# **טכנולוגיות אינטרנט מתקדמות - 61776 (WEB)**

**הגשת פרויקט <Sweat Squad> < A8><3>**

|  |  |
| --- | --- |
| שם חבר.ת הצוות | תז |
| רותם שר | 208608067 |
| סנדרה קניז'ניק | 316181783 |
| אביב כהן | 316576230 |
| יעל קנטר | 315823245 |

< המערכת היא מערכת לניהול לוח אימונים של משתמשים שונים, פר משתמש.

המערכת מאפשרת התחברות באמצעות הזדהות, או לחלופין הרשמה במידה ואין משתמש קיים.

האפליקציה מאפשרת מגוון רחב של אינטראקציות משתמש.

ניתן לצפות בעדכונים אחרונים, לקבוע אימונים שונים ומגוונים, להחליט לגביי מקום האימון במידה ורוצים להתאמן בבית או בחדר הכושר, ניתן לקבוע את זמן האימון ואורכו, ואפילו ניתן להזמין חבר באמצעות אפליקציית הווצאפ המוכרת כאשר כל פרטי האימון הקרוב מפורטים בהזמנה וניתן לבחור למי לשלוח אותה!

המערכת מאפשר לצפות באימוני עבר, אימוני עתיד, לבטל אימון ולהגדיר אימון כבוצע בהצלחה.

ניתן לשנות הגדרות כגון סיסמה, להגדיר יעדי משקל ולצפות בהתקדמות של המתאמן לאורך זמן.

המטרה של המערכת היא לא להחליט על אימונים למתאמן, אלא להחזיר למתאמן את השליטה לידיים, הוא משתמש במערכת כ"יומן אימונים" ומאפשרת למתאמן לנהל **הכל** במקום אחד. >

< מימוש-שמות הטכנולוגיות המרכזיות בכל אחד מהחלקים – Styling - tailwind design

Db – PostgreSQL Database

Backend - node js, experss.js

Frontend- react framework >

<קישור לתיקיית גיט ציבורי : [BACK](https://github.com/AvivFairfield/SweatSquadAPI)  [FRONT](https://github.com/AvivFairfield/A8_React_SweatSquad/tree/main) >

<[קישור לאתר](https://a8-react-sweatsquad.onrender.com)>

<[קישור ל MTW](https://www.morethanwallet.com/app/696)>

תוכן עניינים

[טכנולוגיות אינטרנט מתקדמות - 61776 ( (WEBהגשת פרויקט 3](#_Toc164434845)

הגדרת והקצאת משימות + ממשק בין משתמשים ..............................................3

דרישות ..................................................................................................................4

ארכיטקטורות ודיאגרמות ......................................................................................... 5

פירוט מבנה הDB ....................................................................................................8

**תיק מתכנת**......................................................................................................................................................................9

הצגת תיעוד פונקציות מרכזיות בבק................................................................................9

הצגת תיעוד פונקציות מרכזיות בפרונט..........................................................................14

פירוט ה -API, קישור ל- DB,סביבות מיוחדות, קטעי קוד מיוחדים, פרומפטים בכלי AI...........19

**תיק משתמש**.................................................................................................................................................................20

​ ביצוע התחברות/ רישום לאפליקציה................................................................................20

​ דף הבית במערכת.........................................................................................................20

​ תפריט צד .....................................................................................................................21

​ יצירת אימון עתידי ..........................................................................................................21

​ היסטוריית האימונים .......................................................................................................22

​ מצב התקדמות ..............................................................................................................22

​ עדכון הגדרות משתמש ...................................................................................................22

​ הכרת הנפשות הפועלות ..................................................................................................23

​ התנתקות מהמערכת .......................................................................................................23​

# טכנולוגיות אינטרנט מתקדמות - 61776 ( (WEBהגשת פרויקט

במסגרת הפרויקט נעשה שימוש ב-React למימוש ממשק משתמש עשיר ודינמי, תוך שימוש ב-Hooks וב-Components מותאמים אישית. הבקאנד פותח ב-Node.js תוך שימוש ב-Express.js כמסגרת עבודה וניהול נתונים באמצעות PostgreSQL. העיצוב נעשה באמצעות Tailwind CSS, תוך הקפדה על מודרניות, נגישות.

1. הגדרת והקצאת המשימות בתרגיל זה.

|  |  |  |
| --- | --- | --- |
| **שם חבר הצוות** | **משימות שהוקצו** | **משימות שהושלמו** |
| סנדרה | המרת פרויקט לריאקט  עדכון דיאגרמת USE CASE  תיק משתמש  שקופית הסבר על פונקציה – מצגת להצגה | המרת פרויקט לריאקט  עדכון דיאגרמת USE CASE  שקופית הסבר על פונקציה – מצגת להצגה  תיק משתמש |
| אביב | המרת פרויקט לריאקט  דיאגרמת DB  שקופית הסבר על פונקציה – מצגת להצגה | המרת פרויקט לריאקט  שקופית הסבר על פונקציה – מצגת להצגה  תיק מתכנת |
| יעל | המרת פרויקט לריאקט  דרישות FR/NFR  עדכון ארכיטקטורה של האתר  שקופית אחרונה מצגת להצגה | המרת פרויקט לריאקט  דרישות FR/NFR  עדכון ארכיטקטורה של האתר  שקופית אחרונה מצגת להצגה |
| רותם | המרת פרויקט לריאקט  דיאגרמת תקיות וקבצים  הכנת קובץ פרויקט סופי  שקופית אחרונה מצגת להצגה | המרת פרויקט לריאקט  דיאגרמת תיקיות וקבצים  הכנת קובץ פרויקט סופי  שקופית אחרונה מצגת להצגה |

**הממשק בין חברי הצוות,**

במסגרת הפרוייקט חברי הצוות עבדו באופן קבוע באמצעות ZOOM , ווצאפ. כל שלב בפרוייקט כלל מעבר עצמאי על דרישות כל מטלה ולאחר מכן דיון משותף לגביי הבנת הדרישות וחלוקת העבודה בין חברי הצוות, לעיתים העבודה התבצעה כשכל הצוות יחדיו. לרוב היו עבודות פרטניות או ממקובלות בפלטפורמת ה ZOOM. בנוסף עבדנו בצורה שוטפת עם GITHUB, כל אחד עבד ב VISUAL על BRACH משלו, ולאחר סיום כל חלק העלה אותו לגיט, ומיזג. חלוקת המשימות היה מאוד משמעותית להצלחת הפרוייקט, בעקבות הצורך בניהול הזמנים בסמסטר מורכב, בנוסף התקשורת הבינאישית, העדכונים השוטפים, הכבוד ההדדי והאחריות האישית והקבוצתית היו מאוד משמעותיים ותורמים להצלחת התהליך ולתוצר הסופי.

1. הציגו רשימת דרישות פונקציונליות ולא פונקציונליות

**FR**

1. Users will have the capability to securely register and authenticate their accounts using a username and password.
   1. The app won't let users sign in without correct and existing user name and password, if a user will not stand up to the request the app will alert.
2. Users will be provided with the data about the app managers
   1. From the side bar menu of the app the user will be able to enter a screen called "about" and look at the description of the app managers
3. The system will enable users to update their fitness goals and measurements as needed.
   1. From the side bar menu the user can enter his account settings and update his goal and his weight.
4. The platform will incorporate support for social features, enabling users to engage and interact with each other within the community.
   1. From training selection page the user can invite a friend via WhatsApp
5. The application will support a theme toggle feature, allowing users to switch between dark mode and light mode.
   1. In the user interface, there will be a toggle button accessible from the main navigation.

Users can use this toggle to switch between dark mode, which provides a darker color scheme suitable for low-light environments, and light mode, which offers a brighter color scheme.

**NFR**

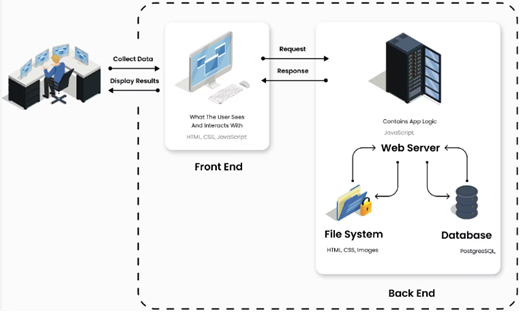
1) Usability: The user interface should be intuitive and easy to navigate.

2) Maintainability: Write clean, modular, and well-documented code to facilitate future updates and maintenance.

3) Integrability: Seamless integration with third-party services such as WhatsApp for workout invitations.

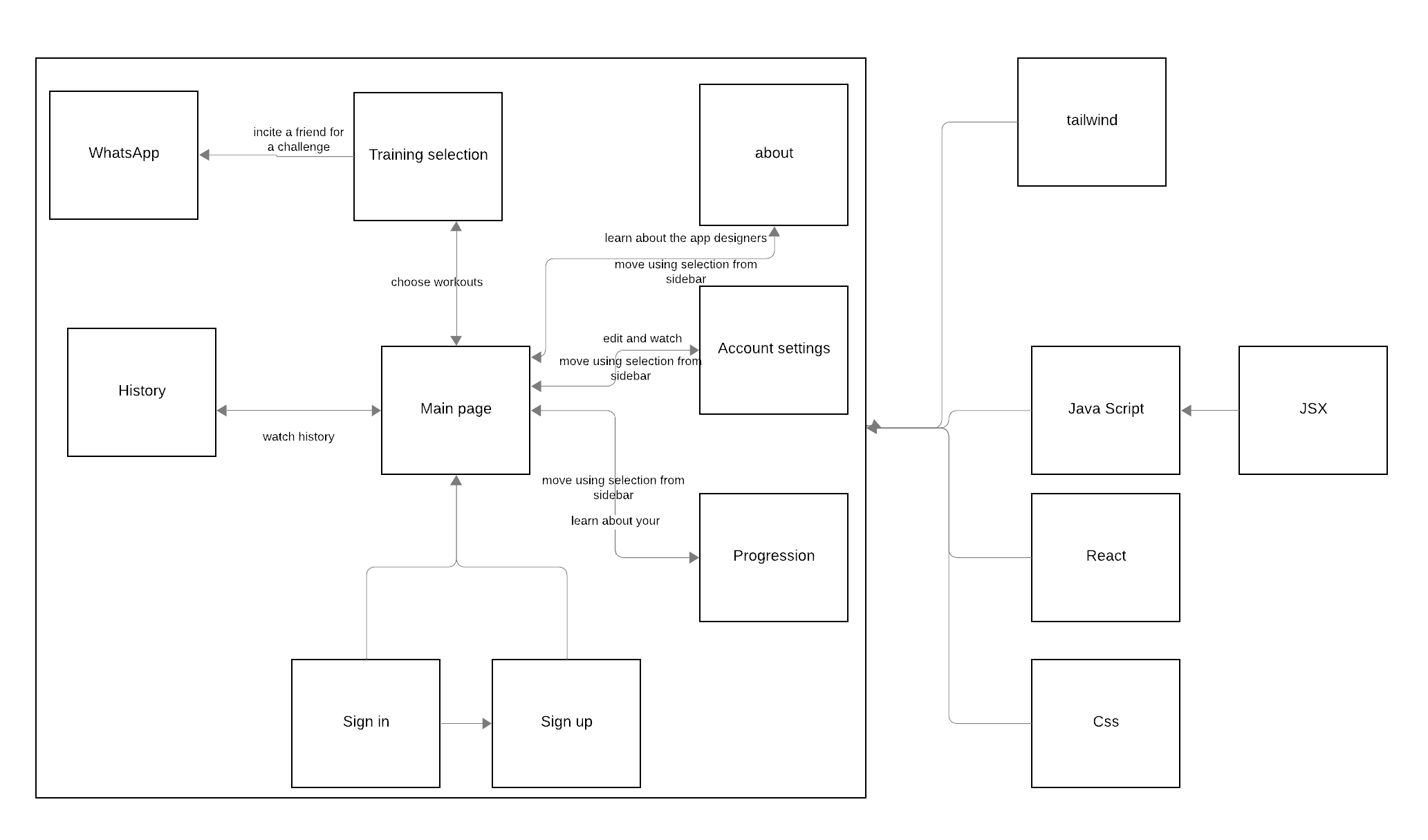
4) Failure management: Provide informative error messages to users

5) Compatibility: Ensure compatibility with a wide range of devices, operating systems, and screen sizes to maximize accessibility and usability for all users.

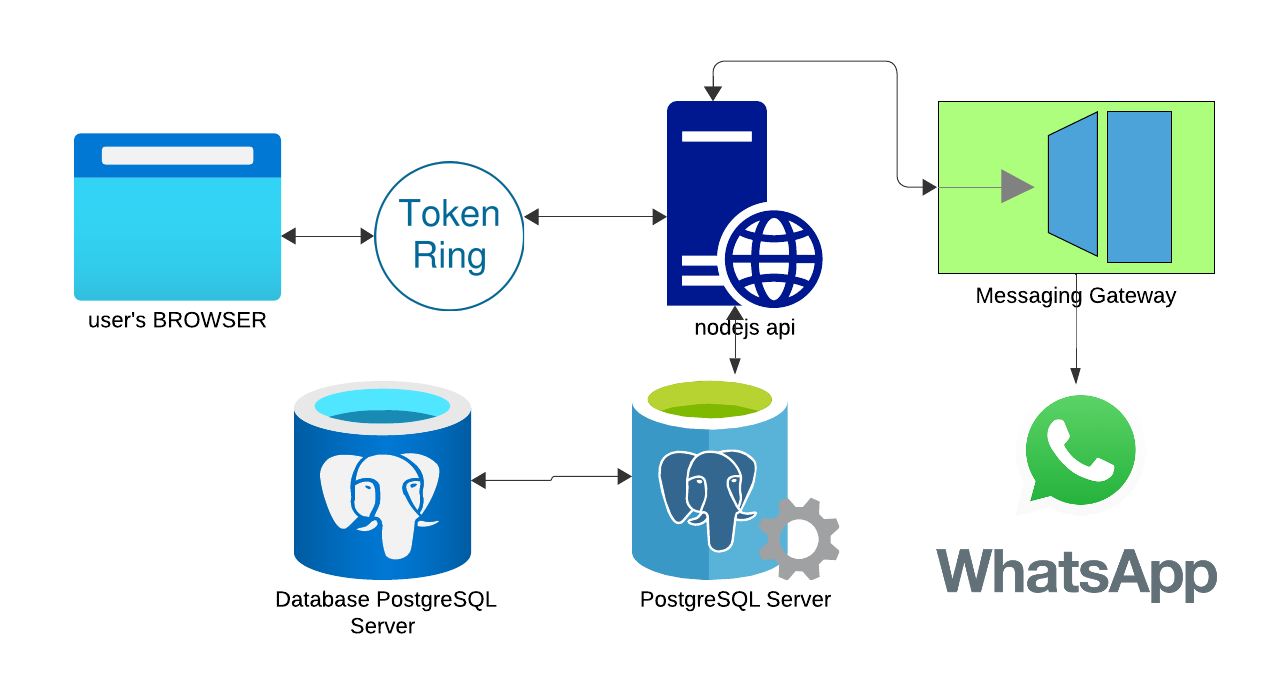
**ארכיטקטורה מעודכנת של האתר**

**HIGH LEVEL VIEW**

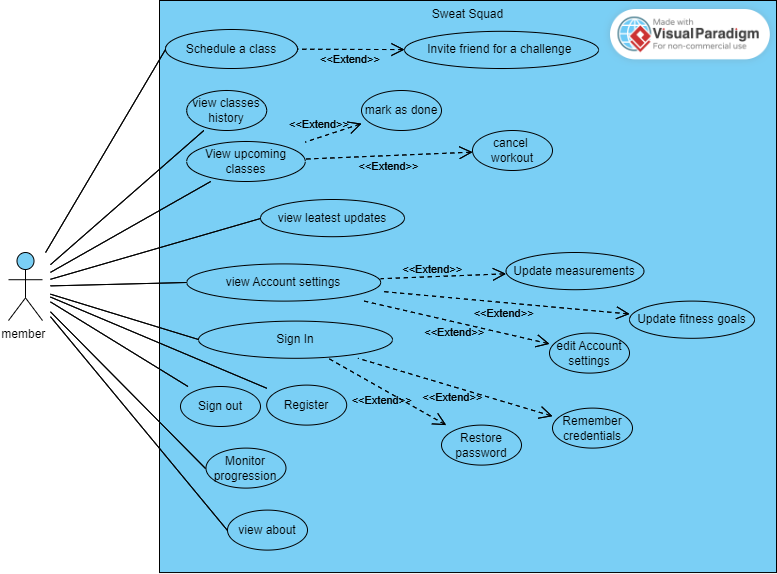
**FRONT END**

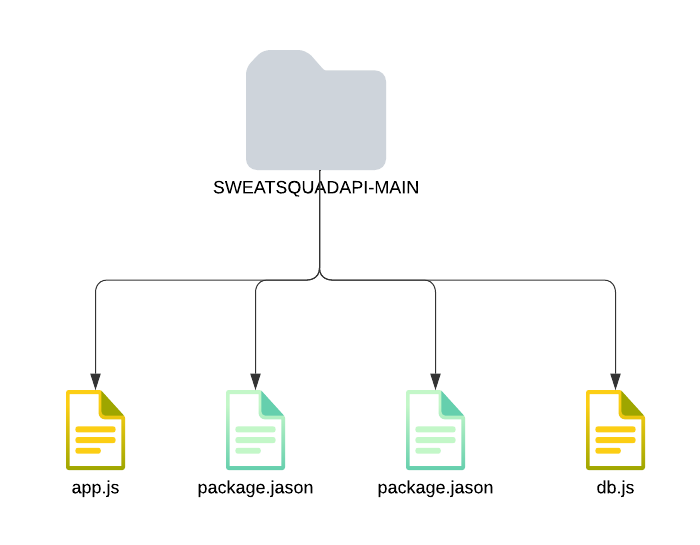
**הערה לגביי שימוש** בSIDEBAR : בכל מסך קיים סיידבר ממנו ניתן להגיע למסכים, home, progression ,account settings, about.

למסכים: TRAINING SELECTION, HISTORY ניתן להגיע רק דרך מסך הMAIN

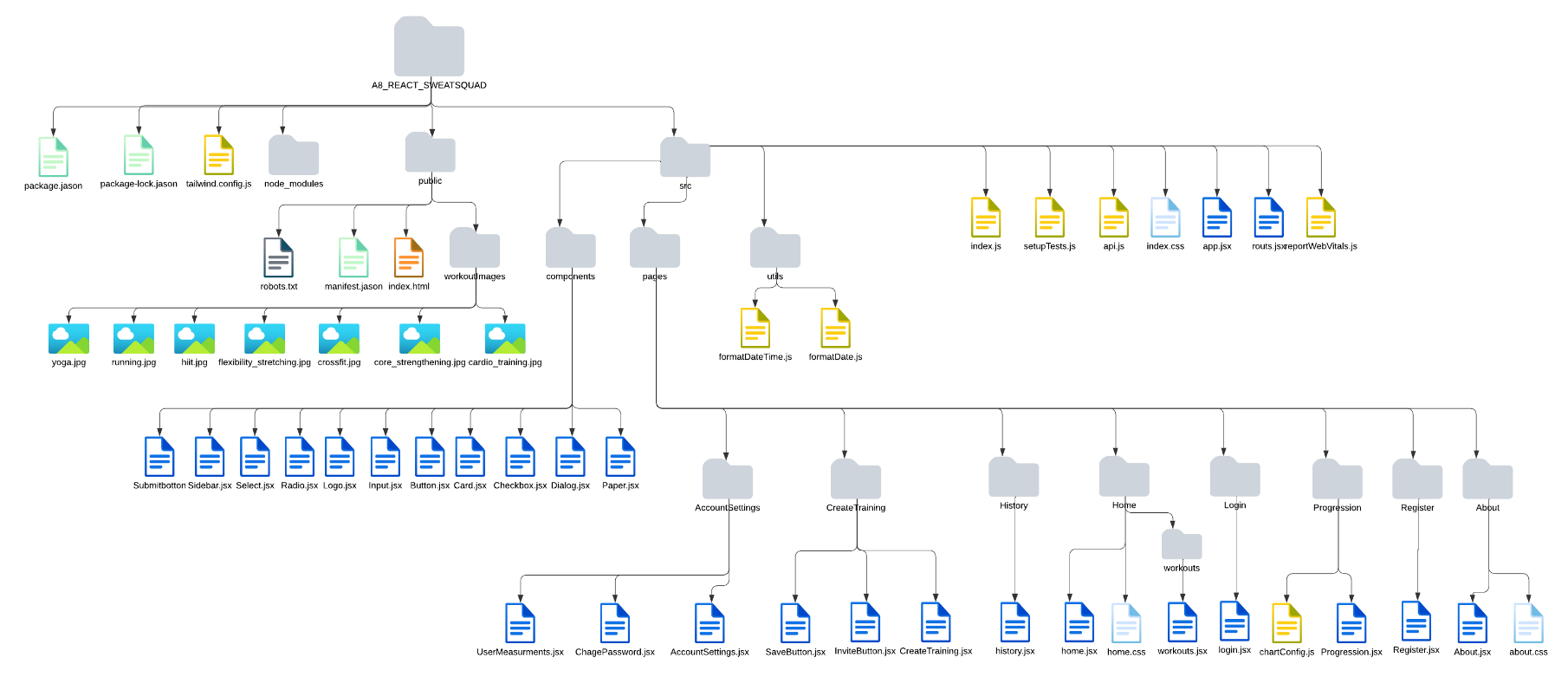
**BACK END**

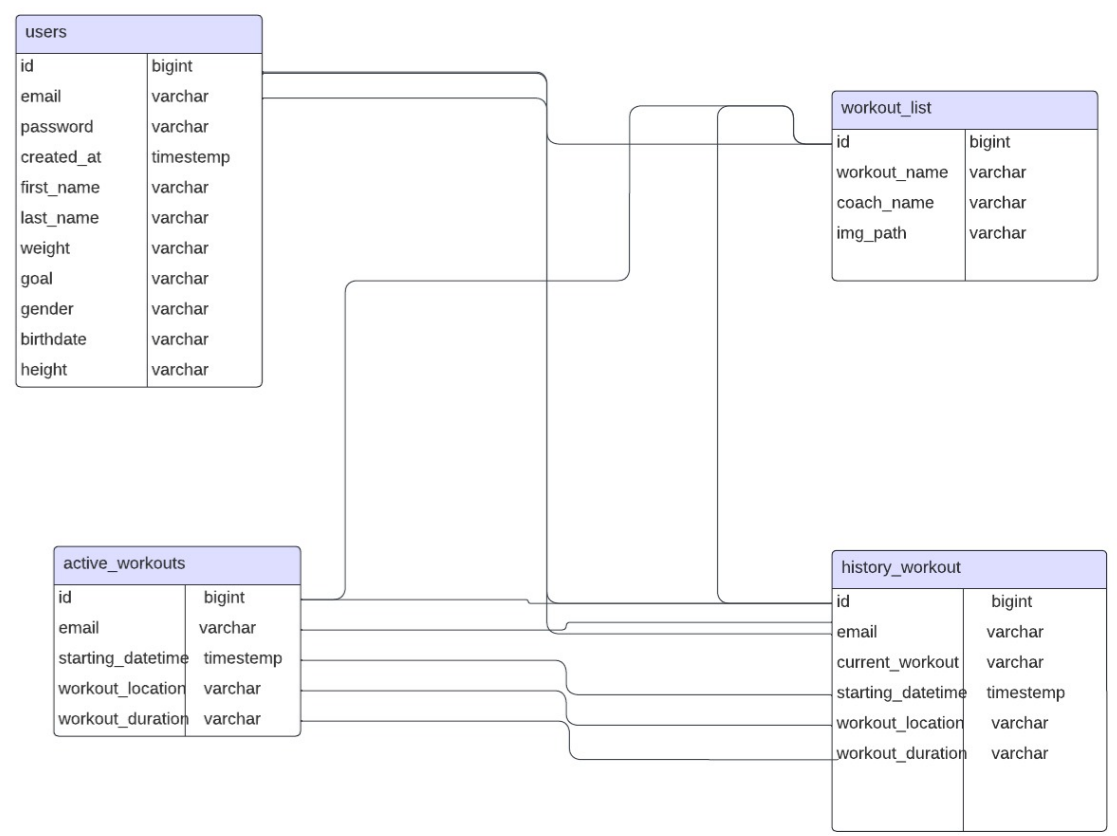
1. דיאגרמת use case :



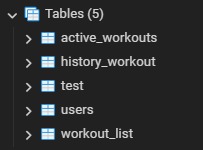
1. מבנה סופי של האתר:   
   א. דיאגרמה המתארת את התיקיות והקבצים השונים.

הפרוייקט מכיל רכיב FRONT ורכיב BACK

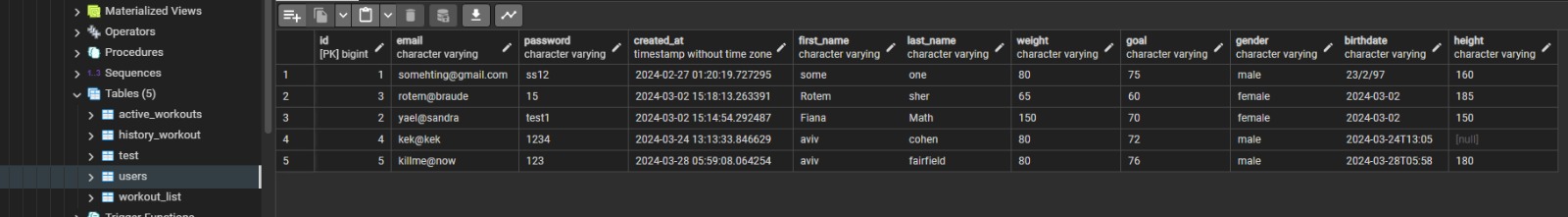
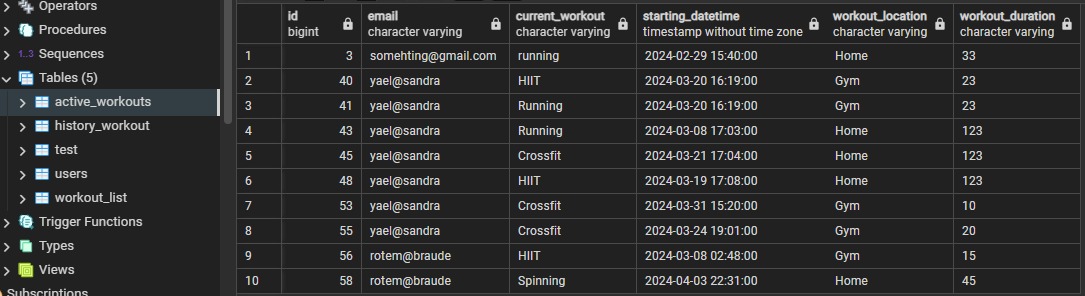


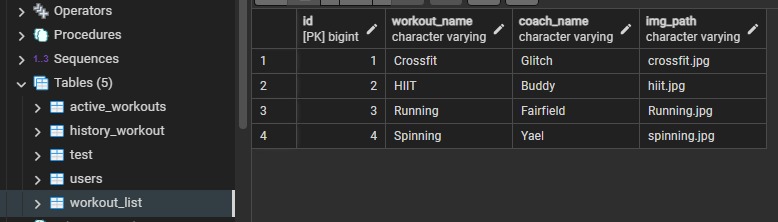


ב. דיאגרמת DB



פריטי מידע מהDB:

* + - 1. טבלת משתמשים : מכילה את פרטי המשתמשים הקיימים במערכת
      2. טבלת שיעורים פעילים : מציגה את השיעורים של המשתמש אותם עוד לא סימן כשהושלמו או מחק
      3. טבלת שיעורים שהושלמו: טבלה המציגה את כל השיעורים אותם סימן המשתמש כשהושלמו
      4. טבלת אימונים לבחירה: טבלה המכילה את כל סוגי האימונים מהם המשתמש יכול לבחור אימון להרשם אליו



תיק מתכנת

**הצגת תיעוד פונקציות מרכזיות**

**בסרבר:**

בסרבר השתמשנו בnode js וexperss.js, והשתמשנו בדאטה בייס "Postgress"

הפונקציה הבאה בקובץ app.js אחראית על התחברות המשתמש למערכת.

השלב הראשון בתוך הפונקציה הוא לחלץ את האימייל והסיסמה מגוף הבקשה. אלו הנתונים שנשלחו על ידי המשתמש המנסה להיכנס. לאחר מכן, בודקים אם הדוא"ל והסיסמה מסופקים בבקשה. אם אחד מהם חסר, מגיבים בהודעת JSON המציינת שיש לספק גם דואר אלקטרוני וגם סיסמה.

הקוד מבצע שאילתות למסד נתונים "postgress",עבור משתמש התואם למייל ולסיסמה שסופקו. לאחר שאילתה למסד הנתונים, הוא בודק אם נמצא משתמש כלשהו שתואם את האישורים שסופקו. אם לא נמצא משתמש, הוא מגיב בהודעת JSON המציינת אימייל או סיסמה לא חוקיים. אם נמצא משתמש, נוצר אסימון (token) אינטרנט של JSON (JWT) בשיטת jwt.sign. אסימון זה כולל את פרטי המשתמש (ID, אימייל, שם פרטי ושם משפחה, סיסמה) ומפתח סודי לחתימה. האסימון מוגדר לפוג בעוד 8 שעות.

לבסוף, השרת מגיב באובייקט JSON הכולל את מצב הפעולה ("הצלחה"), האסימון (( token שנוצר, הודעת קבלת פנים ופרטי המשתמש. בנוסף, קיים בקוד טיפול בשגיאות.

הפעולה כולה עטופה בבלוק try. אם מתרחשת שגיאה במהלך כל שלב בתהליך (לדוגמה, שגיאת מסד נתונים), בלוק ה-catch יתפוס את השגיאה, ירשום אותה ויגיב בהודעת JSON המציינת שהייתה שגיאה בניסיון להתחבר.

//Route to handle user login

app.post("/login", async (req, res) => {

    try {

        const { email, password } = req.body; //Extracting email and password from request body

        console.log(email);

        //Checking if both email and password were provided

        if (!email || !password) {

            res.json({

                message: "You must provide an email & password",

                status: "failed",

            });

            return;

        }

        //Querying the database for a user that matches the provided email and password

        const { rows } = await pool.query(

            "SELECT \* FROM public.users WHERE email = $1 and password = $2",

            [email, password]

        );

        //Check if a user was found

        const userExists = rows.length > 0;

        const user = rows[0]; //The found user

        if (!userExists) {

            res.json({

                message: "Invalid email or password",

                status: "failed",

            });

            return;

        }

        //Generating a JWT for the user

        const token = jwt.sign(

            {

                id: user.id, //Including user details in the token

                email: user.email,

                first\_name: user.first\_name,

                last\_name: user.last\_name,

                password: user.password,

            },

            secretjwt, //Secret key for signing the token

            {

                expiresIn: "8h", //Token expires in 8 hours

            }

        );

        console.log(`token: ${token}`);

        //Responding with the generated token and user details

        res.json({

            status: "success",

            token: token,

            message: `Welcome ${user.first\_name} ${user.last\_name}, Redirecting to homepage...`,

            email: user.email,

            first\_name: user.first\_name,

            last\_name: user.last\_name,

            password: user.password,

        });

    } catch (login\_err) {

        console.log(`login error: ${login\_err}`);

        res.json({

            message: "Error trying to login",

            status: "failed",

        });

        return;

});

הפונקציה הבאה בקובץ app.js אחראית על הרשמת משתמש חדש למערכת.

בתחילת הפונקציה, פרטים שסופקו על ידי המשתמש כגון דואר אלקטרוני, סיסמה, שם מלא, תאריך לידה, משקל, מטרה ומין נשלפים מגוף הבקשה. פרטים אלה נשלחים על ידי צד הלקוח של האפליקציה כאשר משתמש חדש ממלא את טופס ההרשמה. השם המלא שסופק על ידי המשתמש מחולק לשני חלקים כדי להפריד בין השם הפרטי לשם המשפחה. לאחר מכן הפונקציה ממשיכה להכניס את פרטי המשתמש שחולצו לטבלת המשתמשים של מסד הנתונים. אם פעולת מסד הנתונים הצליחה, השרת מגיב עם קוד סטטוס של 200 (אישור) ואובייקט JSON המציין הצלחה והודעת קבלת פנים למשתמש. תגובה זו מעידה על כך שהמידע של המשתמש אוחסן בהצלחה ורישומו הושלם.

//Route to register a new user in the system

app.post("/registeruser", async (req, res) => {

    try {

        //Extract user details from the request body

        const { email, password, fullname, birthdate, weight, goal, gender } =

            req.body;

        full\_name = fullname.split(" ");

        await pool.query(

            `

            INSERT INTO public.users(

                email,

                password,

                first\_name,

                last\_name,

                weight,

                goal,

                gender,

                birthdate)

                VALUES ($1, $2, $3, $4, $5, $6, $7, $8)`,

            [

                email,

                password,

                full\_name[0],

                full\_name[1],

                weight,

                goal,

                gender,

                birthdate,

            ]

        );

        res.status(200).json({

            status: "success",

            message: "Your journey is about to start!",

        });

    } catch (err) {

        // console.log(err);

        res.json({

            status: "failed",

            message: `Invalid input err:${err}`,

        });

    }

});

הפונקציה הבאה בקובץ app.js אחראית על שמירת פרטי האימון עבור משתמש באפליקציה. הפונקציה מתחילה בחילוץ הפרטים הדרושים מגוף הבקשה, דוא"ל, סוג האימון, מיקום, משך זמן ותאריך אימון. פרטים אלו מסופקים על ידי המשתמש כאשר הוא מבקש לשמור אימון. לאחר מכן מוצגת פקודת SQL כדי להכניס את פרטי הפעלת האימון שסופקו לטבלה public.active\_workouts של מסד הנתונים. משתמשים ב-pool.query כדי לבצע פקודה זו, ומעבירים את פרטי הפעלת האימון כפרמטרים לשאילתה. אם פעולת ההוספה מסתיימת בהצלחה, השרת מגיב עם קוד מצב HTTP של 200 (אישור) ואובייקט JSON המציין שהנתונים "נשמרו בהצלחה".

//Route for saving a workout session

app.post("/saveworkout", async (req, res) => {

    try {

        const { email, trainingType, location, duration, workoutDate } =

            req.body;

        console.log(email, trainingType, location, duration, workoutDate);

//sends SQL command to the db to insert the extracted workout session details into the public.active\_workouts table.

        await pool.query(

            `

            INSERT INTO public.active\_workouts(

                email,

                current\_workout,

                workout\_location,

                workout\_duration,

                starting\_datetime)

                VALUES ($1, $2, $3, $4, $5)`,

            [email, trainingType, location, duration, workoutDate]);

        res.status(200).json({

            status: "success",

            message: "Saved Data Succesffully",

        });

    } catch (err) {

        res.json({

            status: "failed",

            message: `Invalid input err:${err}`,

        });

    }

});

**בפרונט:** השתמשנו בreact framework

1. קטע קוד בקובץ app.jsx:

כאשר משתמש פותח את האפליקציה, נבדוק תחילה אם האימייל של המשתמש שמור (מציין שהוא מחובר). אם הוא לא מחובר ומנסה לגשת לדפים אחרים מלבד דפי הכניסה או הרישום, האפליקציה תפנה אותו אוטומטית לדף ההתחברות. לעומת זאת, אם הוא מחובר אך מנסה לבקר בדפי הכניסה או ההרשמה, הוא מופנה לדף הבית.

function App() {

//Hooks for programmatically navigating between routes and accessing the current location.

  const navigate = useNavigate();

  const location = useLocation();

//hecks for an existing "email" key in local storage to determine the user's logged-in status.

  const email = localStorage.getItem("email");

  const isLoggedIn = !!email;

  useEffect(() => {

    //Contains logic for redirecting users based on their authentication

    //state and current location. It ensures that unauthenticated users

    //cannot access certain pages and that logged-in users are redirected from

    //the login/register pages to the home page.

    const currentPageIsRegister = location.pathname.startsWith("/register");

    const currentPageIsLogin = location.pathname.startsWith("/login");

    // If the user is not logged in, redirect to the login page

    if (!isLoggedIn) {

      // If the user is not logged in and in register page, do nothing

      if (currentPageIsRegister) {

        return;

      }

      navigate("/login");

      return;

      // If the user is logged in and in login page, redirect to the home page

    } else if (currentPageIsLogin || currentPageIsRegister) {

      navigate("/");

    }

  }, [isLoggedIn, navigate, location.pathname]);

1. קטע קוד בקובץ createTraining.jsx

הפונקציה getAllWorkouts ממלאת תפקיד מרכזי בקובץ CreateTraining.

פונקציה זו מבוצעת כדי להביא רשימה של כל האימונים הזמינים משרת באמצעות בקשת GET. לאחר מכן, הנתונים שהובאו עוברים טרנספורמציה כך שיתאימו לפורמט הצפוי עבור רכיב ה-Select, המאפשר למשתמשים לבחור את סוג האימון שלהם. מנגנון זה מבטיח שהאפליקציה יכולה להציג רשימה עדכנית של אימונים למשתמש.

 Const getAllWorkouts = async () => {

      try {

        // Sending GET request to fetch all workouts

        const response = await getRequest("/getallworkouts");

        // Processing response if request is successful

        if (response?.status === "success") {

          // Logging response data

          console.log("Success:", response);

          // Mapping received workout data to suitable format for Select component

          const workoutsList =

            response?.workouts?.map((workout) => ({

              value: workout.workout\_name,

              label: workout.workout\_name,

            })) || [];

          // Updating workout options state with received workout data

          setWorkoutsOptions([…defaultWorkoutOptions, …workoutsList]);

        } else {

          // Alerting user if request is unsuccessful

          alert(response?.message);

        }

      } catch (error) {

        // Handling errors during request

        console.error("Error:", error);

      }

    };

1. קטע קוד ב inviteButton.jsx

רכיב InviteButton מאפשר למשתמשים להזמין חברים להצטרף לאתגר האימון שלהם באמצעות WhatsApp. הוא מקבל פרטי אימון , כולל סוג האימון, מיקום, משך ותאריך/שעה. לפני שליחת הזמנה, הוא מוודא שכל השדות הנדרשים ממולאים. אם כן, הוא מעצב את התאריך והשעה של האימון, בונה הודעת הזמנה ויוצר כתובת URL של WhatsApp עם ההודעה. לחיצה על הכפתור פותחת כתובת URL זו בלשונית חדשה, ומאפשרת למשתמש לשלוח את ההזמנה ישירות בוואטסאפ.

export const InviteButton = ({ workoutDetails }) => {

  //Function to handle inviting a friend via WhatsApp

  const inviteFriend = () => {

    const { trainingType, location, duration, workoutDateAndTime } =

      workoutDetails;

  //Check if all fields are filled; if not, alert the user

    if (!trainingType || !location || !duration || !workoutDateAndTime) {

      alert("Please fill in all fields before inviting a friend.");

      return;

    }

    //Formatting the date and time for the workout

    const formattedDateTime = formatDateTime(workoutDateAndTime);

   //Crafting the invitation message with workout details

    const message = `Hey! I'm doing a ${trainingType} workout at the ${location} on ${formattedDateTime}. Wanna join me to my challenge? `;

  //Encoding the message for URL compatibility

    const encodedMessage = encodeURIComponent(message);

  //Generating the WhatsApp URL with the encoded message

    const whatsappUrl = `https://wa.me/?text=${encodedMessage}`;

    window.open(whatsappUrl, "\_blank");

  };

1. קטע קוד בlogin.jsx.

הפונקציה handleSubmitForm ממלאת תפקיד מרכזי בתהליך ההתחברות. הפונקציה מאמתת שממלאים גם מייל וגם סיסמה. אם האימות עובר, הוא ממשיך לשלוח את האישורים הללו לנקודת קצה של ממשק API לכניסה. לאחר קבלת תגובה מוצלחת, הפונקציה מאחסנת את האימייל והשם של המשתמש ב-localStorage ומנווטת את המשתמש לדף הבית. במקרה של ניסיון לא מוצלח, מתריעים בפני המשתמש בהודעת השגיאה.

 //Function to handle form submission

  const handleSubmitForm = async (e) => {

    e.preventDefault();//Prevents the default form submission behavior

    const { email, password } = formState;//Destructures email and password from the form state

    if (!email || !password) return;//Checks if email and password are provided, exits if not

    try {

      //Sends a POST request to the login endpoint with email and password

      const loginResponse = await postRequest("/login", {

        email: email,

        password: password,

      });

       //If the login is successful

      if (loginResponse?.status === "success") {

        //Extracts user information from the response

        const userEmail = email;

        const userFirtname = loginResponse?.first\_name;

        const userLastname = loginResponse?.last\_name;

        //Stores user information in localStorage

        localStorage.setItem("email", userEmail);

        localStorage.setItem("firstname", userFirtname);

        localStorage.setItem("lastname", userLastname);

        navigate("/");//Navigates to the homepage

      } else { //If login is unsuccessful, alerts the user with the message from the response

        alert(loginResponse?.message);

      }

    } catch (error) { //Logs the error to the console if the request fails

      console.error("Error:", error);

    }

  };

5. קטע קוד בhistoty.jsx

בפונקציה הזאת מבוצע איחזור של האימייל של המשתמש מ-localStorage והוא נשלח בבקשת POST לשרת כדי להביא את היסטוריית האימון של המשתמש. לאחר מכן התגובה מעובדת: אם היא מצליחה, התאריך והשעה של כל אימון מעוצבים לקריאות טובה יותר, ומצב הרכיב מתעדכן בנתונים אלו, אשר מוצגים לאחר מכן בטבלה.

useEffect(() => {

    //Asynchronously fetches user's workout history

    const userHistory = async () => {

      const email = localStorage.getItem("email");//Retrieves the user's email from localStorage

      const response = await postRequest("/getuserhistory", {

        email,

      });//Sends a POST request to fetch the workout history

      //If the response status is 'success', process the workouts data

      if (response?.status === "success") {

        const workoutsData =

          response?.workouts?.map((workout) => ({

            ...workout,

            starting\_datetime: formatDateTime(

              new Date(workout.starting\_datetime)

            ),

          })) || [];//Maps over the workouts data to include formatted datetime

        setWorkouts(workoutsData);//Updates the workouts state with the formatted data

      } else {

        alert(response?.message); //Alerts the user if the response status is not 'success'

        console.log(response?.workouts);

      }

    };

    userHistory();

  }, []);

פירוט ה -API, קישור ל- DB,סביבות מיוחדות, קטעי קוד מיוחדים, פרומפטים בכלי AI:

**פירוט ה-API:**

Express.js API, CORS Middleware, JSON Web Token (JWT) API, PostgreSQL Database, whatsapp

**פרומפטים בAI שהשתמשנו בהם:**

Suggest a project timeline for developing a React + Node.js application, including key milestones and deliverables.

Propose a color scheme and typography for a fitness app

Explain how to implement JWT authentication in a Node.js application

1. Provide examples of minimalistic design in web applications.
2. How can I use local storage in a web app to save user settings?
3. What are common errors when connecting a React frontend to a Node.js backend and how can I resolve them?
4. Create a list of resources for learning web development
5. Suggest tools for tracking progress in team software projects
6. Recommend tutorials for beginners on responsive web design.
7. List best practices for writing clean and maintainable CSS

**רפרנסים שנעזרנו בהם:**

<https://react.dev/>

<https://sleekflow.io/blog/whatsapp-link>

<https://expressjs.com/>

<https://jwt.io/introduction>

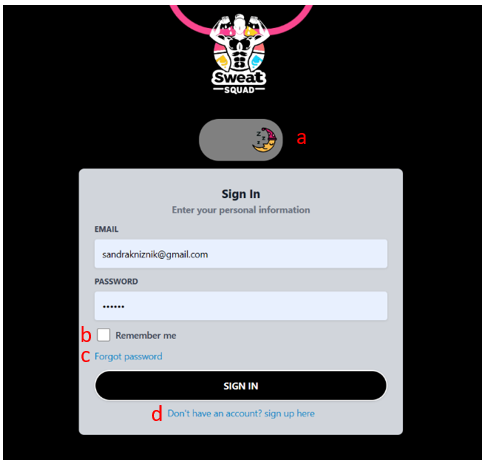
<https://node-postgres.com/>

<https://legacy.reactjs.org/docs/hooks-effect.html>

<https://medium.com/@soulaimaneyh/understanding-cors-7109029406b0>

https://dvmhn07.medium.com/jwt-authentication-in-node-js-a-practical-guide-c8ab1b432a49

**תיק משתמש**

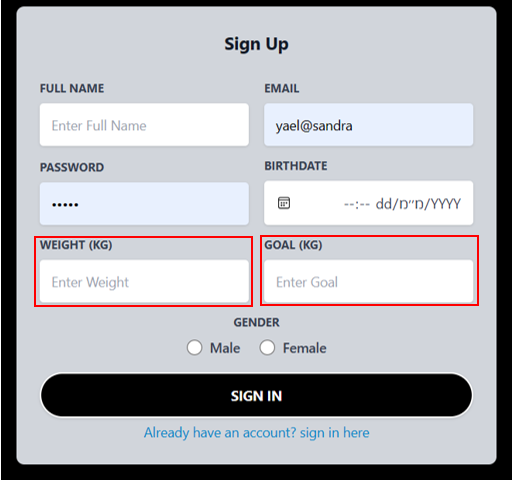
1. **ביצוע התחברות לאפליקציה**

מספר יוזרים אפשריים להתחברות

|  |  |
| --- | --- |
| PASSWORD | EMAIL |
| 123456 | sandrakniznik@gmail.com |
| test1 | yael@sandra |
| ss12 | somehting@gmail.com |

\

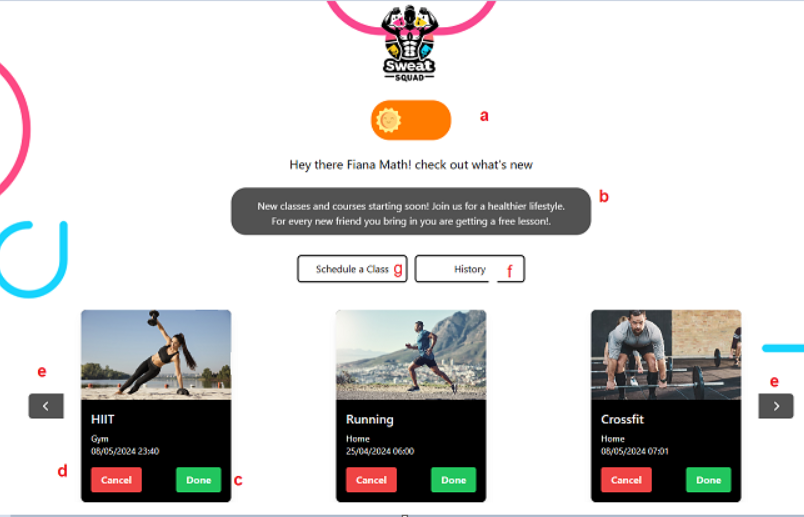
* 1. ניתן להחליף בין DARK ל- LIGHT MODE
  2. זכירת פרטי ההתחברות לפעמים הבאות
  3. במידה ושכחנו את הסיסמה ניתן לעדכן אותה
  4. במידה ואין לנו משתמש ונרצה ליצור חדש קיימת האפשרות



נזין פרטים על מנת ליצור משתמש חדש.

נשים לב להזין את המשקל שלנו ואת משקל היעד.

ונבצע התחברות לאתר



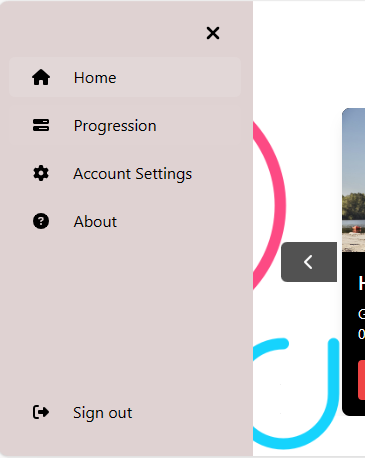
1. **דף הבית במערכת**

לאחר התחברות למערכת, מיידית יפתח מסך הבית

* 1. ניתן להחליף בין DARK ל- LIGHT MODE
  2. מסך הודעות נע המאפשר לראות עדכונים אחרונים ואירועים
  3. סימון אירוע כ"הושלם" -> העברה אוטומטית

של האירוע למסך "HISTORY"

* 1. ביטול אירוע – ימחק מהמערכת ללא זכר
  2. דפדוף בין האימונים העתידיים

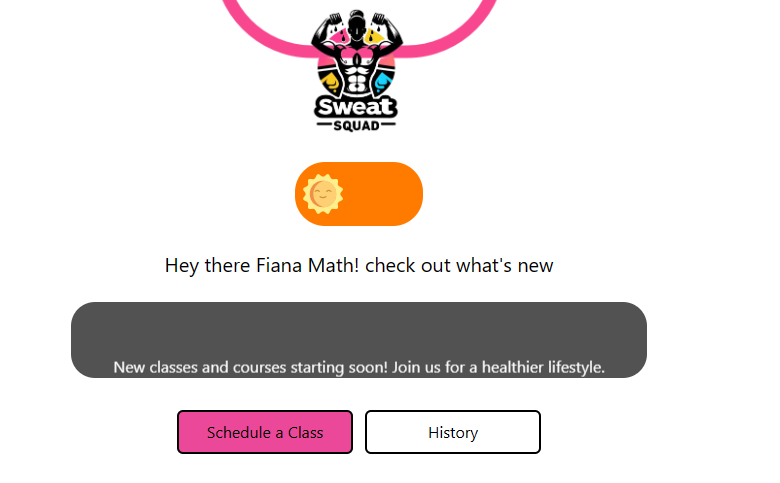
1. **תפריט צד**

לאחר ההתחברות למערכת ניתן לעבור בין מסכי

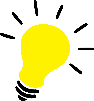
האפליקציה דפדוף בין החלונות בתפריט הצד.

תפריט הצד ימצא בצד השמאלי העליון של כל חלון

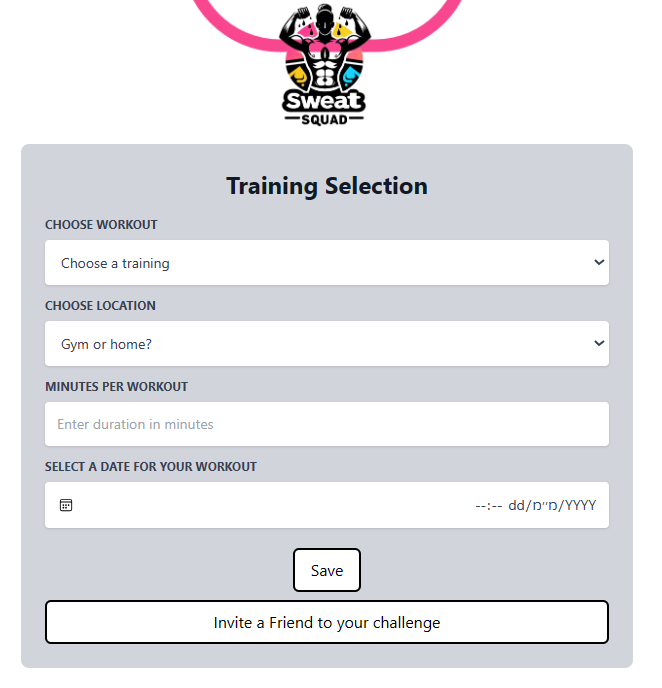
ולחיצה עליו תפתח את התפריט.

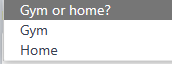
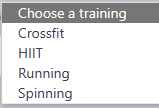
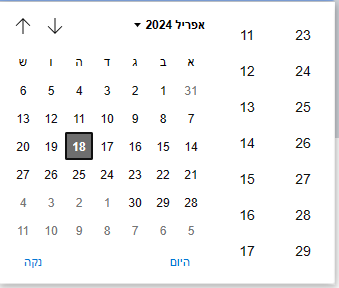
1. **יצירת אימון עתידי**

במסך הבית נבחר בכפתור "SCHEDULE A CLASS"

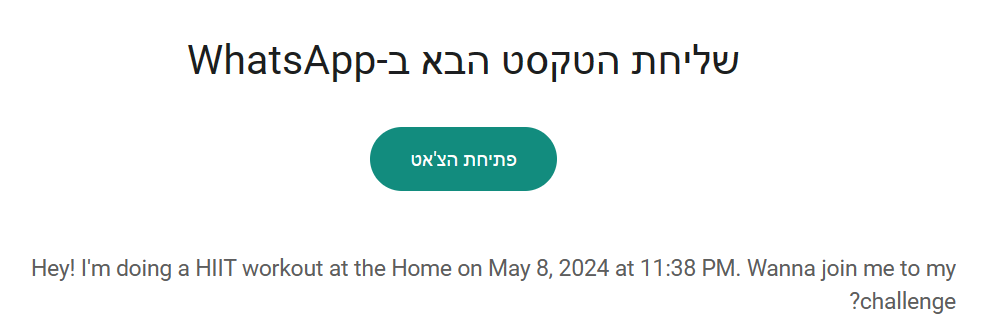


בעת טעינת העמוד יש להמתין כ-20 שניות על מנת

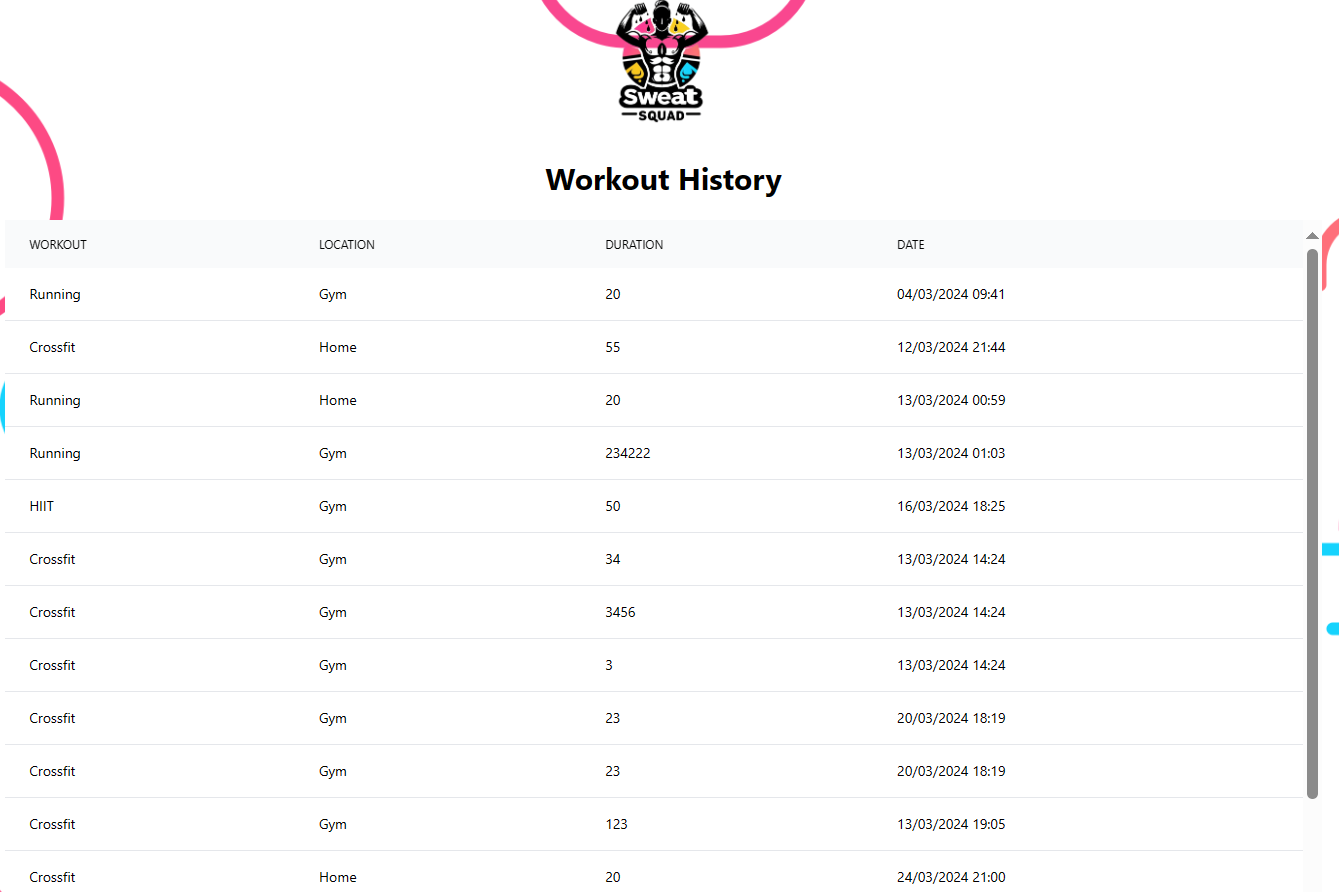


1. בחירת סוג אימון
2. בחירת מיקום האימון
3. בחירת משך האימון
4. בחירת מועד האימון
5. ניתן לשלוח הזמנה לחבר עם פרטי האימון

באמצעות אפליקציית הווצאפ.

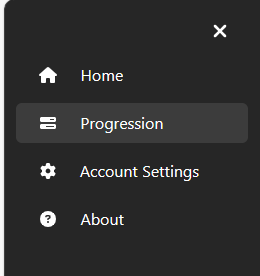


1. **צפייה בהסטוריית האימונים**



ניתן לקבל מידע על סוג האימון,

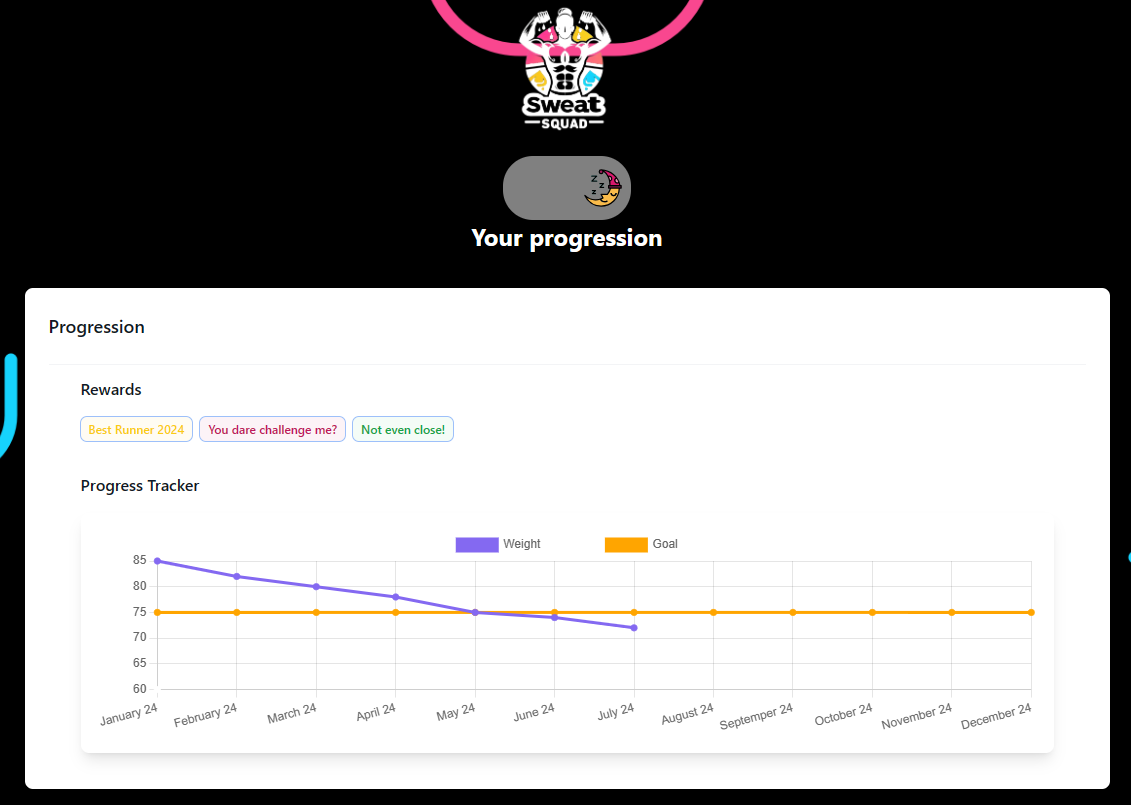
המיקום, משך ותאריך ושעה.

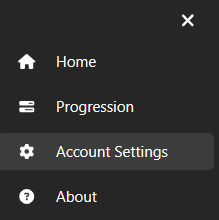
1. **מצב התקדמות**

דרך תפריט הצד, ניתן להגיע למסך התקדמות.

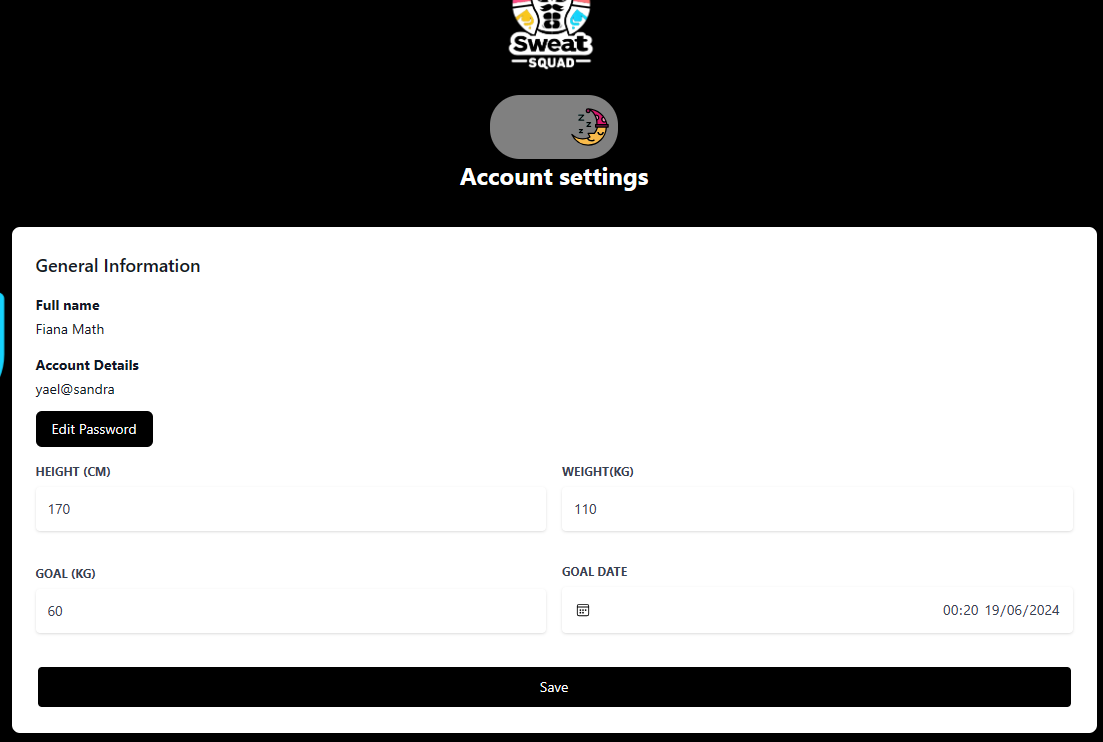
במסך זה המשתמש יכול לבדוק את ההתקדמות

שלו עד כה.



1. **עדכון הגדרות משתמש**

במסך זה המשתמש יכול לעדכן את ההגדרות שלו.



a

b

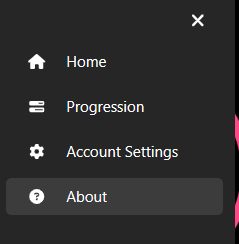
c

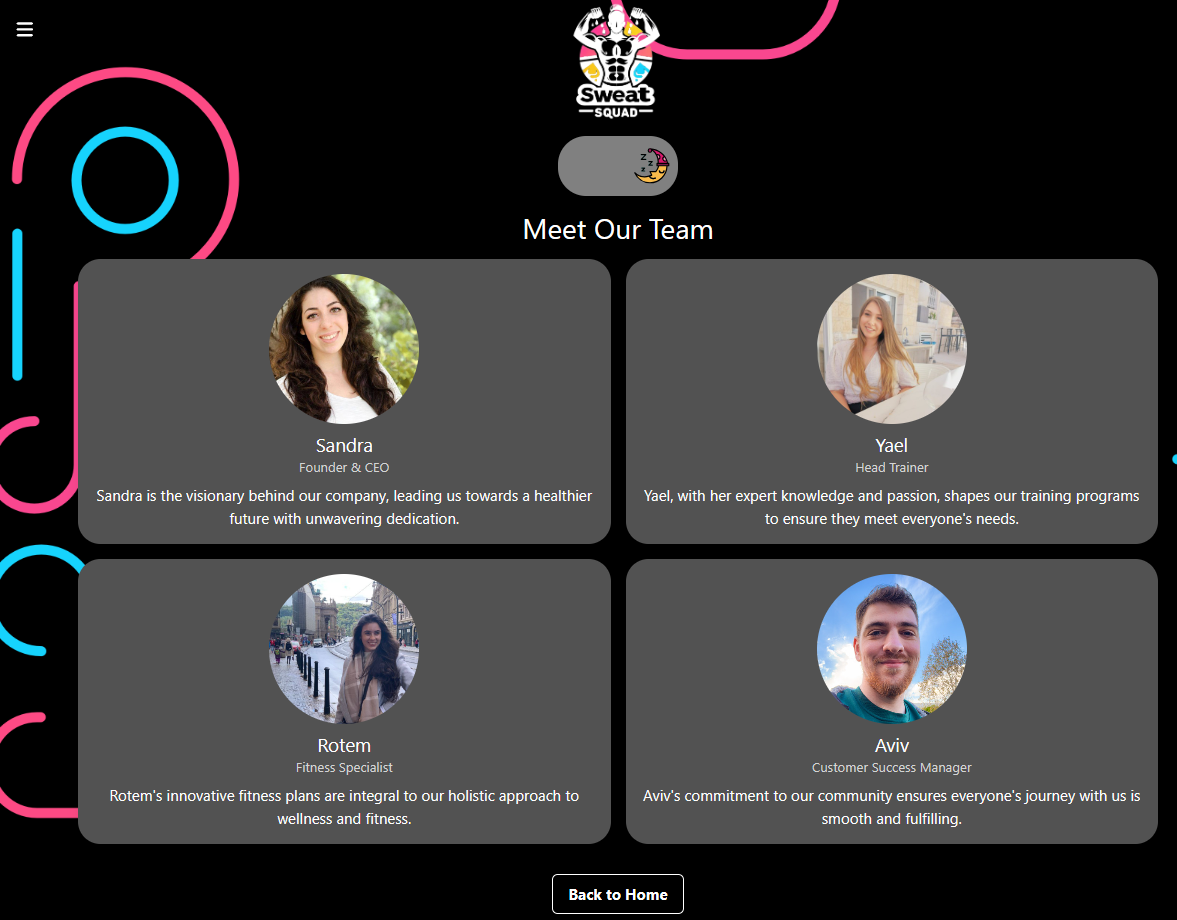
d

e



* 1. עדכון סיסמה
  2. עדכון גובה
  3. עדכון משקל קיים
  4. עדכון משקל יעד
  5. עדכון תאריך יעד

1. **הכרת הנפשות הפועלות**

****

1. **התנתקות מהמערכת**

