

דוח חלק ג

בדוח זה נציג ונפרט על הפונקציונליות של האתר, ננסה להקיף את כל המידע הנדרש ומהלך העבודה שלנו בחלק זה. נציין כי כדי לבחון את הפונקציות בצורה המלאה אנו כמובן ממליצים להשתמש באתר ☺

1. מבנה תיקיות הפרויקט:

- מבנה תיקיות הפרויקט הוא במבנה המומלץ:
- א. תיקיית DB ובתוכה כל הקבצים הכוללים של DB כמו קובץ ה Config, connection , פונקציות CRUD, בנוסף לכך כל קבצי ה-csv המכילים את המידע של הטבלאות שלנו.
 - ב. תיקיית static ובתוכה כל הקבצים הגרפיים- תיקיית CSS ובה העיצובים , תיקיית JS של צד לקוח. תיקיית תמונות שממנה נלקחות תמונות לתצוגה באתר שלנו.
 - ג. תיקיית view ובתוכה כל קבצי Pug שלנו.

בתוך תיקיית Root הראשית אנו מיקמנו את שלל התיקיות לעיל ובנוסף לכך את קובץ Index.js אשר מריץ את האתר שלנו.

2. ביצענו טיפול בבקשות לקוח על ידי routing:

הגדרנו את הבית שלנו להיות Home-page, בנוסף קיימים מספר routings נוספים. קיימים לנו 3 עמודים אשר נבנים בעקבות הנתונים בDB.

2.1 עמוד של classes : בעמוד זה אנו נרצה להציג את כל האימונים הקיימים לנו בסטודיו בפירוט של מה סוג האימון כולל , ותמונה שמתארת את האימון.

אנו נבצע קריאה של טבלת trainings ונשלח את המידע של טבלה זו באמצעות משתנה אשר נקרא בצד הלקוח.

2.2 עמוד של coaches: בעמוד זה נרצה להציג את כל המאמנים הקיימים לנו בסטודיו בפירוט מה ההתמחות שלהם . אנו נבצע קריאה של טבלת coaches ונשלח את המידע של טבלה זו באמצעות משתנה הנקרא בצד הלקוח.

2.3 עמוד של טבלת אימונים: בטבלת האימונים ניתן לראות את כלל האימונים שהוזנו למערכת עבור השבוע הנוכחי. לכל אימון יש יום, שעה, סוג.

קיימים עמודים נוספים אשר נפרט עליהם בסעיף הבא.

צילומי מסך, routing:

```
shahar bar
app.get('/', (req :... , res :Response<ResBody, LocalsObj> ) => {
  res.render( view: 'Home-page');
});

shahar bar
app.get('/Sign_in', (req :... , res :Response<ResBody, LocalsObj> ) => {
  res.render( view: 'SignIn');
});

shahar bar
app.get('/Sign_up', (req :... , res :Response<ResBody, LocalsObj> ) => {
  res.render( view: 'Sign-Up');
});

shahar bar
app.get('/classes', (req :... , res :Response<ResBody, LocalsObj> ) => {
  const Q1 = 'select * from Trainings';
  mysql.query(Q1, callback: (err :QueryError| null , mysqlres : (RowDataPacket[][]) | ... )=>{
    if (err) throw err;
    res.render( view: 'Training', options: {V1 : mysqlres})
  })
});
```

```
shahar bar
app.get('/Registration', (req :... , res :Response<ResBody, LocalsObj> ) => {
  const q1='select * from';
  res.render( view: 'Registration for classes');
});

shahar bar
app.get('/Contact', (req :... , res :Response<ResBody, LocalsObj> ) => {
  res.render( view: 'Contact-us');
});

shahar bar
app.get('/Table-classes', (req :... , res :Response<ResBody, LocalsObj> ) => {
  const Q1 = 'SELECT time_training, day_training, GROUP_CONCAT(CONCAT(day_training, '\ - \', Coacher_name, '\ - \', training) SEPARATOR '\n\n') AS details FROM table_of_training_no_user';
  mysql.query(Q1, callback: (err :QueryError| null , mysqlres : (RowDataPacket[][]) | ... ) => {
    if (err) throw err;
    const scheduleData = mysqlres.map(row => {
      return {
        time_training: row.time_training,
        day_training: row.day_training,
        details: row.details ? row.details.split('\n') : []
      };
    });
    console.log(scheduleData[0].details);
    res.render( view: 'Table of classes.pug', options: { Data: scheduleData });
  });
});

shahar bar
app.get('/Review', (req :... , res :Response<ResBody, LocalsObj> ) => {
  res.render( view: 'Review');
});
```

```
shahar bar
app.get('/Coach', (req :... , res :Response<ResBody, LocalsObj> ) => {
  const Q1 = 'select * from Coachers';
  mysql.query(Q1, callback: (err :QueryError| null , mysqlres : (RowDataPacket[][]) | ... )=>{
    if (err) throw err;
    res.render( view: 'Coach', options: {V1 : mysqlres})
  })
});
```

3. חיבור לבסיס הנתונים:

3.1 יצרנו קובץ המכיל את המידע האישי שלנו לחיבור לDB שלנו הלוקאלי. ניתן לשנות את קובץ זה בהתאם לחיבור הלוקאלי של כל משתמש.

```
module.exports = {  
  HOST: "localhost",  
  USER: "root",  
  PASSWORD: '20121995',  
  DB: "web"  
};
```

3.2 יצרנו קובץ בשם DB אשר מכיל בתוכו את החיבוריות וייצוא החיבור לשרת.

```
const mysql = require('mysql2');  
const dbConfig = require('./db.config.js');  
const connection = mysql.createConnection( config: {  
  host: dbConfig.HOST,  
  user: dbConfig.USER,  
  password: dbConfig.PASSWORD,  
  database: dbConfig.DB  
});  
  
shahar bar  
connection.connect( callback: error => {  
  if (error) throw error;  
  console.log("Successfully connected to the database.");  
});  
  
module.exports = connection;
```

3.3 בנינו routing עבור יצירת כל הטבלאות בבסיס נתונים:

טבלת users:

שם השדה	סוג הנתונים
תעודת זהות	Int
אימייל	String
שם פרטי	String
שם משפחה	String
סיסמא	String
דרגת מצב בריאותי	Int

טבלת הרשמה לאימונים:

שם השדה	סוג הנתונים
תעודת זהות של המתאמן שנרשם לאימון	Int
תעודת זהות של המאמן	Int
שם השיעור	String
שעה של השיעור	time
ביקורת על האימון	String
תאריך של השיעור	Date

טבלת מאמן:

שם השדה	סוג הנתונים
תעודת זהות	Int
שם מאמן	String
תיאור	bigint
תמונה	String

טבלת אימונים:

שם השדה	סוג הנתונים
שם השיעור	String
שעה של השיעור	time
תאריך של השיעור	Date
שם המאמן	String

3.3 בנוסף לכך ביצענו עמוד init אשר בו ניתן לאתחל את השרת עם הטבלאות.
בעמוד זה ניתן לבצע: Create, Insert, Drop בלחיצה אחת עבור כלל הטבלאות.

על מנת להיכנס לעמוד זה יש להיכנס לנתיב הבא:

<http://localhost:3000/init>

להלן תמונת העמוד:

Initialization database: Tables and Data

Please use these 3 links for: CREATE, INSERT, DROP
Enjoy our website

[CREATE tables](#)

[INSERT data](#)

[DROP tables](#)



בעמוד זה אנו נשתמש בפונקציות:

```
app.get ( '/createAll', CreateDB.create_AllTables );  
app.get ( '/dropAll', CreateDB.drop_Alltables );  
app.get ( '/insertAll', CreateDB.inset_Alldata );
```

3.6 בנספחים מתוארים כל פונקציות ה-CRUD שהוספנו.

4. מימוש טפסים:

טופס sign in - בטופס זה בדקנו את הנתונים המוכנסים על ידי הלקוח – שם משתמש וסיסמא, ובדקנו האם הנתונים הללו קיימים בDB כדי לדעת האם להכניסו למערכת בתור לקוח רשום. לאחר מכן שמרנו את הID שלו בcookie על מנת שיהיה לנו לפעולות עתידיות בהמשך (הרשמה לאימונים, פרופיל אישי וכו').

טופס sign up - בטופס זה אנו מבצעים בדיקות ולידציה עבור הכנסת הנתונים של משתמש DB. אנו יצרנו חוקים עבור הקלט והודענו ללקוח בעזרת התראות כיצד להזין את הפרטים האישיים שלו באופן תקין בעת הרשמה לאתר. (ראה נספחים)

טופס registration for classes - בטופס זה בדקנו איזה מהאימונים פנויים לרישום- כלומר האם קיימים אימונים בהם יש מקום להרשמה. אם באימון מסויים יש מקום לרישום, האימון יוצג ברשימה נגללת למשתמש לשם הרשמה. בעת הרשמה מוכנס הID של cookie המחובר לאתר.

טופס user-profile - בטופס זה אנו מציגים את פרטי הלקוח המחובר לאתר: שם, מצב בריאותי ורשימת האימונים אליהם המשתמש רשום בשבוע הקרוב. בנוסף, בטופס זה יש כפתורים נוספים: ניתוק המשתמש (ומחיקת cookie), יצירת קשר ובקשה לשינוי פרטים אישיים. את כלל המידע המותאם למשתמש שלפנו בעזרת שאילתות הנעזרות בCookie.

טופס table of classes - אנו הצגנו למשתמשים את כל האימונים הפתוחים השבוע מסודרים לפי יום האימון ושעת האימון בסדר עולה. בכל שורה מתואר האימון בעזרת שם המאמן וסוג האימון אותו הוא מעביר. את נתונים אלו אנו שולפים בעזרת שאילתות.

טופס home-page - בדף הבית למעשה שינינו את nav בהתאם לcookie הקיים. כאשר המשתמש עוד לא מחובר לאתר יוצג לו כפתור sign up/sign in. לאחר שהמשתמש ביצע התחברות, כפתור sign in/sign up לא יוצג, ובמקומו יוצג לו כפתור הפרופיל האישי my profile.

טופס trainings - בטופס זה משכנו את הנתונים הקיימים לנו מהDB של האימונים, ויצרנו טופס דינאמי אשר משתנה בהתאם לנתונים בDB. הצגנו את כל האימונים הקיימים כרגע בסטודיו בעזרת שם האימון, מה הוא כולל, איזה תמונה תואמת.

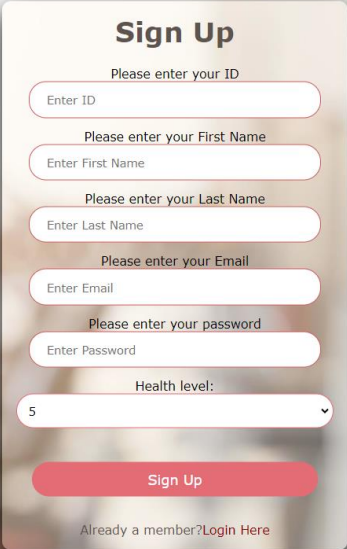
טופס coaches - בטופס זה משכנו את הנתונים הקיימים לנו מהDB של המאמנים, ויצרנו טופס דינאמי אשר משתנה בהתאם לנתונים בDB. הצגנו את כל המאמנים הקיימים כרגע בסטודיו בעזרת שם המאמן, באילו אימונים הוא מתמחה, איזה תמונה תואמת.

טופס review - בטופס זה משכנו מהDB את האימונים הרשומים של המשתמש הספציפי שמחובר לאתר. לאחר מכן הצגנו לו רשימת גלילה של אימונים אלו להם הוא יוכל לכתוב ביקורת. כתיבת הביקורת תעדכן את טבלת האימונים ותעדכן את הביקורת שנתן המשתמש על האימון הספציפי הזה.

טופס contact us - בטופס זה נתנו למשתמש אפשרות לכתוב לנו במגוון מקרים, כאשר כתיבה בטופס זה שולח לנו מייל ישירות עם הפניה ונוכל לטפל בפניה שלו.

5. מימוש פונקציונליות עיבוד מידע:

יצירת משתמש חדש דרך sign up:



Sign Up

Please enter your ID
Enter ID

Please enter your First Name
Enter First Name

Please enter your Last Name
Enter Last Name

Please enter your Email
Enter Email

Please enter your password
Enter Password

Health level:
5

Sign Up

Already a member? [Login Here](#)

בפונקציה זו אנו מושכים את המידע שהוזן מהמשתמש ל־Form על מנת להכניס את נתונים אלו ל־DB. לאחר מכן אנחנו מעדכנים את המשתנה הגלובלי להיות עם ערכים אלו. אנו שולחים הודעות למשתמש בהתאם לשגיאות / הצלחה בהרשמה. ביצירת היוזר קיימת תקשורת עם הלקוח- היות וקיים JS של ואלידציה בצד לקוח. צד שרת:

```
const createNewUser = (req,res)=> {
  //validate that body is not empty
  if (!req.body) {
    res.status(400).send({message: "content can not be empty"});
    return;
  }
  //insert data to json
  const NewUser = {
    "id": req.body.ID,
    "firstname": req.body.FirstName,
    "lastname": req.body.LastName,
    "email": req.body.Email,
    "password": req.body.password,
    "Health": req.body.Health
  };

  mysql.query( sql: "INSERT INTO users SET ?", NewUser, {callback: (err :QueryError|null , mysqlres :{RowDataPacket[]|...}) => {
    if (err) {
      console.log("error: ", err);
      res.status(400).json({ message: "error in creating user: " + err }); // Change send to json
      return;
    }
    else {
      user.id = NewUser.id;
      user.firstname = NewUser.firstname;
      user.lastname = NewUser.lastname;
      user.password = NewUser.password;
      user.email = NewUser.email;
      user.Health = NewUser.Health;
      console.log("created new user: ", { id: mysqlres.insertId });
      res.status(201).json({ message: "User created", id: mysqlres.insertId });
    }
  });
};
```

בצד השרת יש עמוד PUG בשם sign up :

- בעמוד זה הלקוח נרשם לאתר על ידי הזנת ID ולידי – של 8 ספרות
- שם פרטי ושם משפחה על ידי הזנה של אותיות בלבד ולא של ספרות.
- אימייל על ידי הזנה של אימייל תקין .
- סיסמא על ידי הזנה של אות גדולה , אות קטנה , סימן , 6 ספרות.
- מצב בריאותי על ידי משיכה של הערך שהוזן בselect.

בדיקת הוולידציות מתבצעת בJS:

```
// Get references to the form and the inputs
const form = document.querySelector( selectors: '.sign-in-form');
const passwordInput = document.querySelector( selectors: '#password');
const firstNameInput = document.querySelector( selectors: '#FirstName');
const lastNameInput = document.querySelector( selectors: '#LastName');
const emailInput = document.querySelector( selectors: '#Email');
const idInput = document.querySelector( selectors: '#ID');
//const healthInput = document.getElementById('HealthLevel');
const healthInput = document.querySelector( selectors: '#HealthLevel');
// Define regular expressions for validation
var passwordRegex = /^(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&*])(?=.*[0-9]).{8,}$/;
var nameRegex = /^[a-zA-Z]+$/;
var emailRegex = /^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,}$/;

// Add event listeners for form submission
shahar bar
form.addEventListener( type: 'submit', listener: function(event: Event) {
  // Prevent the form from submitting if validation fails
  event.preventDefault();

  const password = passwordInput.value.trim();
  const firstname = firstNameInput.value.trim();
  const lastname = lastNameInput.value.trim();
  const email = emailInput.value.trim();
  const id = idInput.value.trim();
  const health = healthInput.options[healthInput.selectedIndex].value;
```

הגדרנו רגקסים שונים על מנת להגדיר את חוקיות הולדיות של כל שדה.

לכל שדה קיימת פונקציית ולדיות:

```
Usage shahar bar
function validatePassword() {
  const passwordValue = passwordInput.value;
  if (!passwordRegex.test(passwordValue)) {
    alert('Password must contain at least 8 characters, including at least one uppercase letter, one lowercase letter, one number, and one special character.');
```

```
Usage shahar bar
function validateFirstName() {
  const nameValue = firstNameInput.value;
  if (!nameRegex.test(nameValue)) {
    alert('Please enter a valid first name using only letters.');
```

```
Usage shahar bar
function validateLastName() {
  const nameValue = lastNameInput.value;
  if (!nameRegex.test(nameValue)) {
    alert('Please enter a valid last name using only letters.');
```



```

1 usage  👤 shahar bar
function validateEmail() {
  const emailValue = emailInput.value;
  if (!emailRegex.test(emailValue)) {
    alert('Please enter a valid email address.');
```

🔍

```

    emailInput.focus();
    return false;
  }
  return true;
}

1 usage  👤 shahar bar
function validateID() {
  //const id = document.getElementById("ID").value;
  const id = document.querySelector(selectors: '#ID').value;
  const regexID = /\b\d{8}\b/;
```

🔍

```

  if (!regexID.test(id)) {
    alert('Please enter a valid ID');
```

🔍

```

    return false;
  }
  return true;
}

```

```

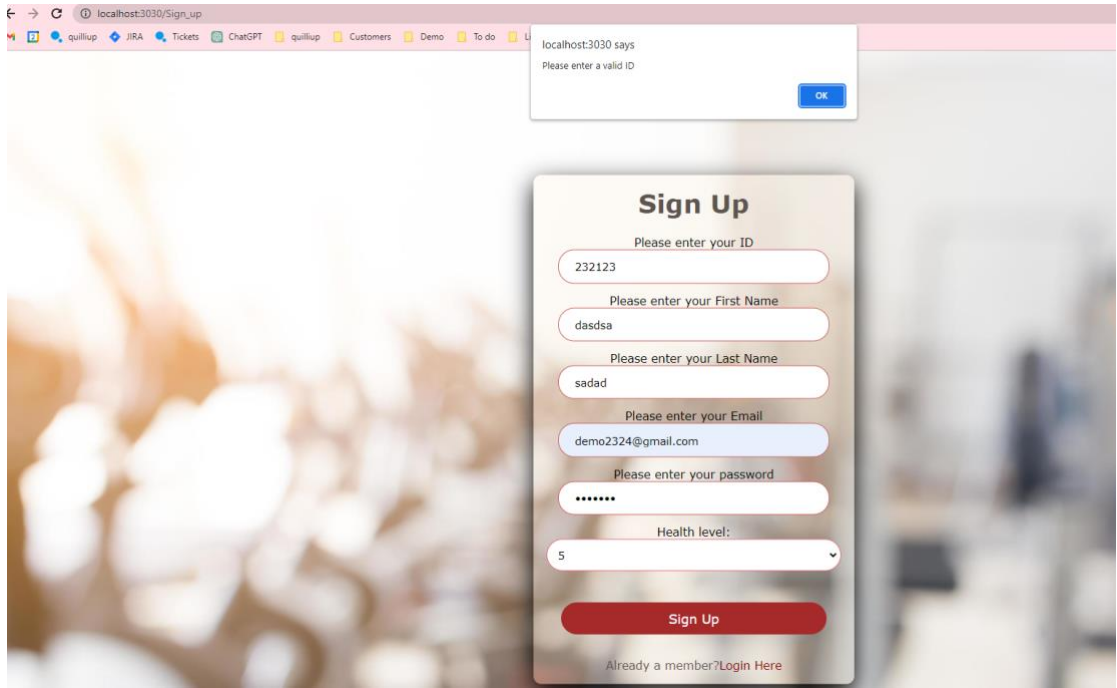
if (!validateID() || !validateFirstName() || !validateLastName() || !validateEmail() || !validatePassword()) {
  return;
}
// Form a user data object
const userData = {
  ID: id,
  Email: email,
  password: password,
  FirstName: firstname,
  LastName: lastname,
  Health: health
};

// Send data to server
fetch( input: "/createNewCustomer", init {
  method: "POST",
  headers: {
    "Content-Type": "application/json",
  },
  body: JSON.stringify(userData),
}) .Promise<Response>
  .then((res : Response) => res.json()) .Promise<any>
  .then((data) => {
    // console.log("Response received from /Sign_up");
    console.log(data);

    alert(
      "Your account has been created "
    );
    window.location.href = "/";
    passwordInput.value = "";
    firstNameInput.value = "";
    lastNameInput.value = "";
    emailInput.value = "";
    idInput.value = "";
  });

```

לאחר שהמידע עובר את כלל הבדיקות ונמצא כתקין אנו נזין אותו אל בסיס הנתונים. לאחר מכן תוצג הודעה שהרישום בוצע והשרת יחזיר אותנו אל אמור ה־sign לטובת התחברות.



נספחים :

3.שאילתות ליצירת הDB:

```
const user = {
  "id": '',
  "firstname": '',
  "lastname": '',
  "email": '',
  "password": '',
  "Health": ''
};

const UserIDcookie={
  "value": '',
};
```

יצירת משתנים גלובלים של הקוקי ושל הנתונים של היוזר שמתחבר למערכת/ נרשם למערכת.

3.1 יצירת יוזר חדש- דרך sign up:

בפונקציה זו אנו מושכים את המידע שהוזן מהמשתמש לForm על מנת להכניס את נתונים אלו לDB.

לאחר מכן אנחנו מעדכנים את המשתנה הגלובלי להיות עם ערכים אלו.

אנו שולחים הודעות למשתמש בהתאם לשגיאות / הצלחה בהרשמה.

ביצירת היוזר קיימת תקשורת עם הלקוח- היות וקיים JS של ואלידציה בצד לקוח.

```
const createNewUser = (req,res)=> {
  //validate that body is not empty
  if (!req.body) {
    res.status(400).send({message: "content can not be empty"});
    return;
  }
  //insert data to json
  const NewUser = {
    "id": req.body.ID,
    "firstname": req.body.FirstName,
    "lastname": req.body.LastName,
    "email": req.body.Email,
    "password": req.body.password,
    "Health": req.body.Health
  };

  mysql.query( `sql: "INSERT INTO users SET ?", NewUser, {callback: (err :QueryError | null, mysqlres : (RowDataPacket[] | ... ) => {
    if (err) {
      console.log("error: ", err);
      res.status(400).json({ message: "error in creating user: " + err }); // Change send to json
      return;
    }
    else {
      user.id = NewUser.id;
      user.firstname = NewUser.firstname;
      user.lastname = NewUser.lastname;
      user.password = NewUser.password;
      user.email = NewUser.email;
      user.Health = NewUser.Health;
      console.log("created new user: ", { id: mysqlres.insertId });
      res.status(201).json({ message: "User created", id: mysqlres.insertId });
    }
  })` );
};
```

צד שרת:

צד לקוח:

עמוד PUG: sign up :

בעמוד זה הלקוח נרשם לאתר על ידי הזנת ID ולידי – של 8 ספרות
שם פרטי ושם משפחה על ידי הזנה של אותיות בלבד ולא של ספרות.
אימייל על ידי הזנה של אימייל תקין.
סיסמא על ידי הזנה של אות גדולה , אות קטנה , סימן , 6 ספרות.
מצב בריאותי על ידי משיכה של הערך שהוזן בselect.

בדיקת הווילדות מתבצעת בJS:

```
// Get references to the form and the inputs
const form = document.querySelector(selectors: '.sign-in-form');
const passwordInput = document.querySelector(selectors: '#password');
const firstNameInput = document.querySelector(selectors: '#FirstName');
const lastNameInput = document.querySelector(selectors: '#LastName');
const emailInput = document.querySelector(selectors: '#Email');
const idInput = document.querySelector(selectors: '#ID');
//const healthInput = document.getElementById('HealthLevel');
const healthInput = document.querySelector(selectors: '#HealthLevel');
// Define regular expressions for validation
var passwordRegex = /^(?=.*\d)(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#$%^&*]).{8,}$/;
var nameRegex = /^[a-zA-Z]+$/;
var emailRegex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

// Add event listeners for form submission
shahar bar
form.addEventListener( type: 'submit', listener: function(event: Event) {
  // Prevent the form from submitting if validation fails
  event.preventDefault();

  const password = passwordInput.value.trim();
  const firstname = firstNameInput.value.trim();
  const lastname = lastNameInput.value.trim();
  const email = emailInput.value.trim();
  const id = idInput.value.trim();
  const health = healthInput.options[healthInput.selectedIndex].value;
```

הגדרנו רגקסים שונים על מנת להגדיר את חוקיות הולדות של כל שדה.

לכל שדה קיימת פונקצית ולדות:

```
Usage shahar bar
function validatePassword() {
  const passwordValue = passwordInput.value;
  if (!passwordRegex.test(passwordValue)) {
    alert('Password must contain at least 8 characters, including at least one uppercase letter, one lowercase letter, one number, and one special character. ');
    passwordInput.focus();
    return false;
  }
  return true;
}

// Validate name inputs
Usage shahar bar
function validateFirstName() {
  const nameValue = firstNameInput.value;
  if (!nameRegex.test(nameValue)) {
    alert('Please enter a valid first name using only letters. ');
    firstNameInput.focus();
    return false;
  }
  return true;
}

Usage shahar bar
function validateLastName() {
  const nameValue = lastNameInput.value;
  if (!nameRegex.test(nameValue)) {
    alert('Please enter a valid last name using only letters. ');
    lastNameInput.focus();
    return false;
  }
  return true;
}
```

```

1 usage  👤 shahar bar
function validateEmail() {
    const emailValue = emailInput.value;
    if (!emailRegex.test(emailValue)) {
        alert('Please enter a valid email address. ');
        emailInput.focus();
        return false;
    }
    return true;
}

1 usage  👤 shahar bar
function validateID() {
    //const id = document.getElementById("ID").value;
    const id = document.querySelector(selectors: '#ID').value;
    const regexID = /\b\d{8}\b/;
    if (!regexID.test(id)) {
        alert('Please enter a valid ID');
        return false;
    }
    return true;
}

```

```

if (!validateID() || !validateFirstName() || !validateLastName() || !validateEmail() || !validatePassword()) {
    return;
}
// Form a user data object
const userData = {
    ID: id,
    Email: email,
    password: password,
    FirstName: firstname,
    LastName: lastname,
    Health: health
};

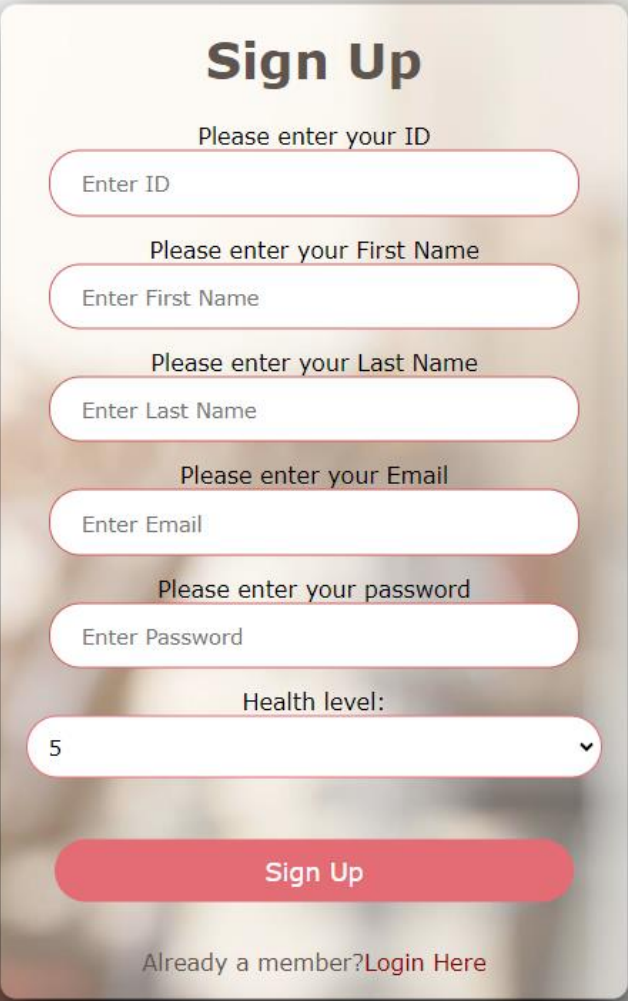
// Send data to server
fetch( input: "/createNewCustomer", init: {
    method: "POST",
    headers: {
        "Content-Type": "application/json",
    },
    body: JSON.stringify(userData),
}) Promise<Response>
    .then((res : Response) => res.json()) Promise<any>
    .then((data) => {
//      console.log("Response received from /Sign_up");
      console.log(data);

      alert(
        "Your account has been created "
      );
      window.location.href = "/";
      passwordInput.value = "";
      firstNameInput.value = "";
      lastNameInput.value = "";
      emailInput.value = "";
      idInput.value = "";

```

אנו נשלח את כל המידע לשרת לאחר בדיקת הולדיות להזנה בתוך בסיס הנתונים.

לבסוף השרת יחזיר לנו תשובה ויעביר אותנו לעמוד sign in של המערכת.



Sign Up

Please enter your ID

Enter ID

Please enter your First Name

Enter First Name

Please enter your Last Name

Enter Last Name

Please enter your Email

Enter Email

Please enter your password

Enter Password

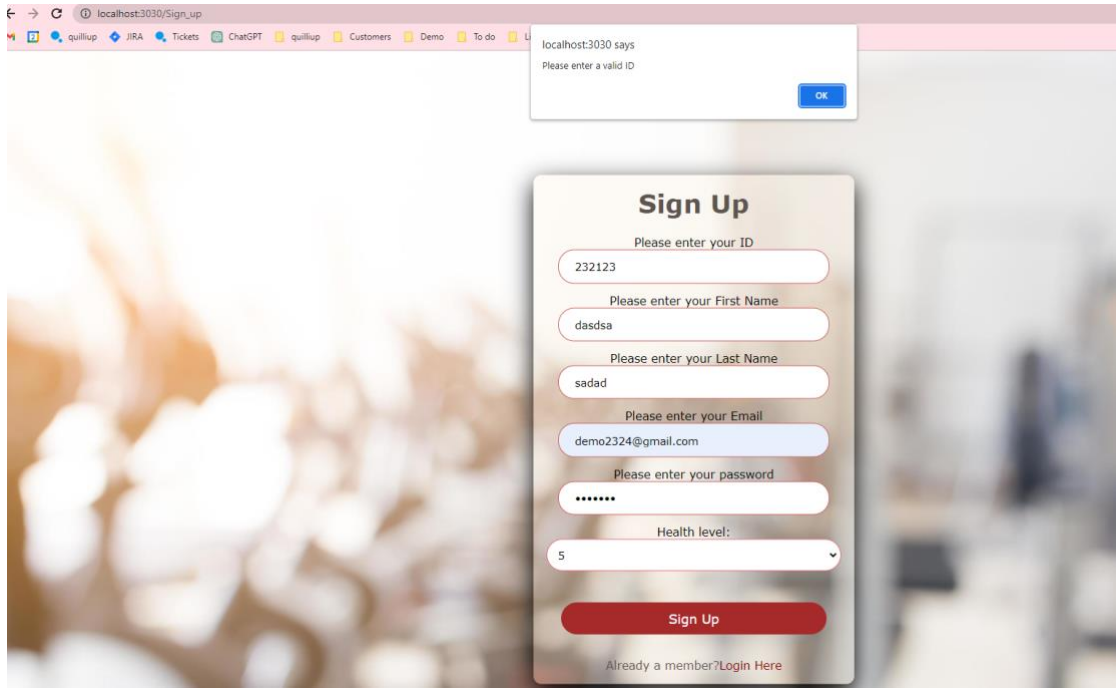
Health level:

5

Sign Up

Already a member? [Login Here](#)

הערות ישלחו כמו alert ללקוח:



3.2 מציאת יוזר קיים – כניסה דרך sign in

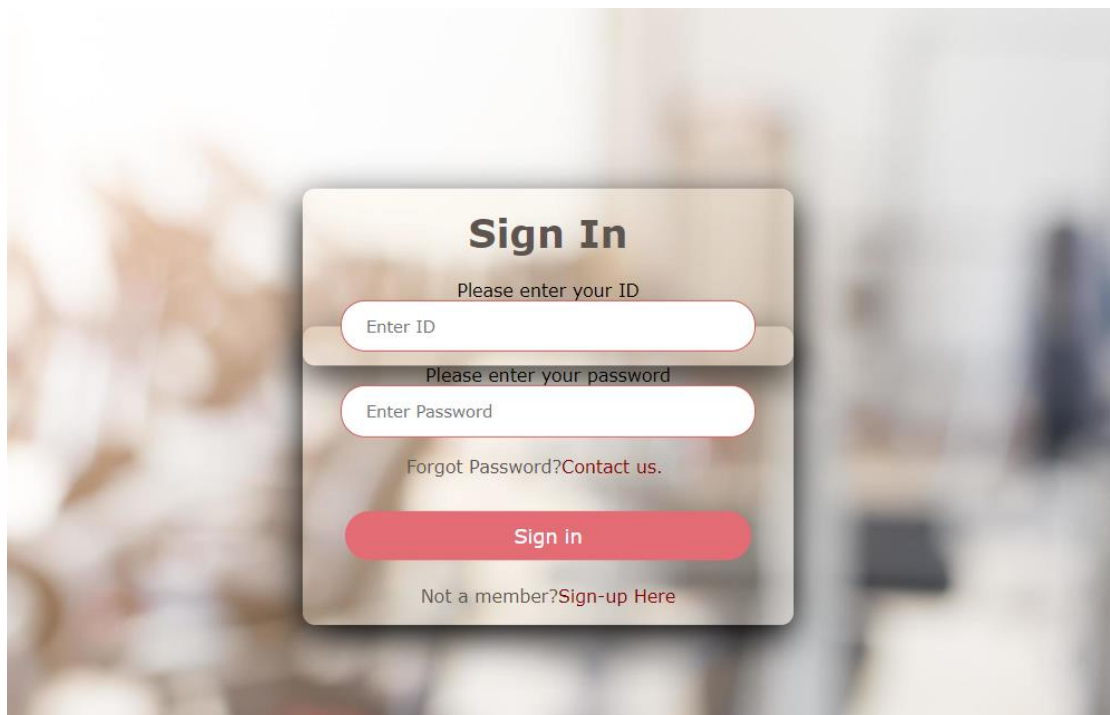
```
const FindUser = (req,res)=> {
  //validate that body is not empty
  if (!req.body) {
    res.status(400).send({message: "content can not be empty"});
    return;
  }
  //insert data to json
  const User_exists = {
    "id": req.body.id,
    "password": req.body.password
  };
  mysql.query( `SELECT * FROM users where id = ?`, User_exists.id, (err:QueryError|null, mysqlres:(RowDataPacket[])) => {
    if (err) {
      console.log("error in getting all user " + err);
      res.status(400).send({message: "error in getting all user " + err});
      return;
    }
    if (mysqlres.length == 0) {
      res.send('<script>alert("The ID is not correct! please try again "); window.location.href = "/Sign_in";</script>');
      return;
    }
    else {
      mysql.query( `SELECT * FROM users where password like ?`, values: User_exists.password + "%", (err:QueryError|null, mysqlres:(RowDataPacket[])) => {
        if (err) {
          console.log("error in getting all user " + err);
          res.status(400).send({message: "error in getting all user " + err});
          return;
        }
        if (mysqlres.length == 0) {
          res.send('<script>alert("The password is not correct! please try again "); window.location.href = "/Sign_in";</script>');
          //res.render('Loginscreen', {var1: "You successfully sign in"});
          return;
        }
        else {
          console.log("User sign in successfully");
          //res.render('Loginscreen', {var1: "You successfully sign in"});
          user.id=User_exists.id;
          user.id=User_exists.id;
          user.firstname=User_exists.firstname;
          user.lastname=User_exists.lastname;
          user.password=User_exists.password;
          user.email=User_exists.email;
          user.Health=User_exists.Health;
          res.cookie ('UserID',req.body.ID);
          UserIDcookie.value=req.cookies.UserID;
          console.log(req.cookies);
          res.send('<script>alert("The user is sign in "); window.location.href = "/";</script>');
          return;
        }
      });
    }
  });
};
```

אנו נרצה כאשר לקוח מתחבר למערכת דרך עמוד sign in שאנו נבדוק האם הוא קיים בDB.

לכן נבדוק את הid שלו ואת הסיסמא האם היא תואמת למידע הקיים לנו בDB כדי לדעת איפה להודיע לו שקיימת הבעיה.

עמוד sign in -PUG:

```
.wrapper
  h1 Sign In
  form.sign-in-form(action='/CheckUser' method='post')
    label(for='ID') Please enter your ID
    input#ID(type='tel' name='ID' placeholder='Enter ID' required='')
    label(for='password') Please enter your password
    input#password(type='password' name='password' placeholder='Enter Password' required='')
  .recoverPass
    | Forgot Password?
    a(href='/Contact') Contact us.
  button Sign in
  .member
    | Not a member?
    a(href='/Sign_up')
    | Sign-up Here
```



5.2 פונקציית myprofile = עמוד פרופיל אישי

```
const MyProfile(req,res)=> {
  //validate that body is not empty
  if (!req.body) {
    res.status(400).send({message: "content can not be empty"});
    return;
  }
  else {
    const query1 = "SELECT u.firstname, u.lastname, u.health, t.time_training, t.day_training, t.training, t.Coacher_name FROM users u LEFT JOIN table_of_training t ON u.id = t.user_id";
    mysql.query(query1, user_id, callback: (err: QueryError | null, mysqlres: RowDataPacket[]) => {
      if (err) {
        console.log("error in getting all user " + err);
        res.status(400).send({message: "error in getting all user " + err});
        return;
      }
      else {
        console.log(req.cookies);
        const query2 = "SELECT u.firstname, u.lastname, u.health, t.time_training, t.day_training, t.training, t.Coacher_name FROM users u LEFT JOIN table_of_training t ON u.id = t.user_id";
        mysql.query(query2, req.cookies?.UserID, callback: (err: QueryError | null, mysqlres1: RowDataPacket[]) => {
          if (err) {
            console.log(req.cookies?.UserID);
            console.log("error in getting all user " + err);
            res.status(400).send({message: "error in getting all user " + err});
            return;
          }
          else {
            console.log(mysqlres1[0]);
            const trainings = mysqlres1.map(training => {
              return {
                "time": training.time_training,
                "day": training.day_training,
                "coacherName": training.Coacher_name,
                "trainingName": training.training
              };
            });
            console.log(trainings);
            res.render('User profile', {
              variable1: 'Hi ' + mysqlres1[0].firstname + ' Welcome back',
              variable3: 'Health Level: ' + mysqlres1[0].Health,
              variable4: 'Your upcoming lessons:',
              variable5: trainings
            });
            return;
          }
        });
      }
    });
  }
};


const MyProfile = (req, res) => {
  //validate that body is not empty
  if (!req.body) {
    res.status(400).send({message: "content can not be empty"});
    return;
  }
  else {
    const query1 = "SELECT u.firstname, u.lastname, u.health, t.time_training, t.day_training, t.training, t.Coacher_name FROM users u LEFT JOIN table_of_training t ON u.id = t.user_id";
    mysql.query(query1, user_id, callback: (err: QueryError | null, mysqlres: RowDataPacket[]) => {
      if (err) {
        console.log("error in getting all user " + err);
        res.status(400).send({message: "error in getting all user " + err});
        return;
      }
      else {
        console.log(req.cookies);
        const query2 = "SELECT u.firstname, u.lastname, u.health, t.time_training, t.day_training, t.training, t.Coacher_name FROM users u LEFT JOIN table_of_training t ON u.id = t.user_id";
        mysql.query(query2, req.cookies?.UserID, callback: (err: QueryError | null, mysqlres1: RowDataPacket[]) => {
          if (err) {
            console.log(req.cookies?.UserID);
            console.log("error in getting all user " + err);
            res.status(400).send({message: "error in getting all user " + err});
            return;
          }
          else {
            console.log(mysqlres1[0]);
            const trainings = mysqlres1.map(training => {
              return {
                "time": training.time_training,
                "day": training.day_training,
                "coacherName": training.Coacher_name,
                "trainingName": training.training
              };
            });
            console.log(trainings);
            res.render('User profile', {
              variable1: 'Hi ' + mysqlres1[0].firstname + ' Welcome back',
              variable3: 'Health Level: ' + mysqlres1[0].Health,
              variable4: 'Your upcoming lessons:',
              variable5: trainings
            });
            return;
          }
        });
      }
    });
  }
};
```


בפרופיל האישי אנו בעצם שולפים את כל האימונים שהיוזר המחובר בקוקי נרשם אליהם בשבוע הקרוב ופרטים אישיים נוספים עליו.

אנו נשלח את המשתנים האלה לעמוד הפאג user-profile בצד הקליינט :

```
label.user-icon
i.fa-solid.fa-user
.profile-items
h1= variable1
p= variable3
h2= variable4
table
  tr
    th
    th
    th
    th
  each training in variable5
    tr
      td #{training.time}
      td #{training.day}
      td #{training.coacherName}
      td #{training.trainingName}
.button-container
button.button(id='review-button') Review Class
button.button(id='update-profile-button') Update Profile
button.button(id='cancel-membership-button') Cancel Membership
```

ניתן לראות כי משתנים אלו יהיו בעמוד הפרופיל האישי ויודפסו למשתמש כל האימונים אליהם הוא נרשם. אם לא יהיו למשתמש אימונים אליהם הוא נרשם הטבלה תהיה ריקה.

 [HOME](#) [FLOW COACHES](#) [FLOW CLASSES](#) [FLOW WEEK SCHEDULE](#) [CONTACT US](#) [MY PROFILE](#)



Hi shahar Welcome back


Health Level: 5

Your upcoming lessons:

09:00:00 Sunday	Matar Gershoni	Functional training
19:00:00 Sunday	Katya Kalinoviz	Functional training
20:00:00 Sunday	Amitay Gelre	Pilates Training
09:00:00 Monday	Yoni cohen	TRX Training
10:00:00 Monday	Matar Gershoni	Yoga Training
19:00:00 Monday	Matar Gershoni	TRX Training
18:00:00 Tuesday	Yafit halewer	Zumba Training
10:00:00 Friday	Katya Kalinoviz	Yoga Training

[Review Class](#) [Update Profile](#) [Cancel Membership](#)

Find us at:
1 Tse'elim Street
Meitar

Navigate now:


Contact us:
Shahar: 0526555351
Omer: 0545620696

5.3 פונקציית mytraining review – כתיבת ביקורת לכל האימונים שנרשמת אליהם:

```
const mytrainings_review = (req, res) => {
  //validate that body is not empty
  if (!req.body) {
    res.status(400).send({message: 'content can not be empty'});
    return;
  }
  else {
    const query2 = `SELECT u.firstname, u.lastname, u.health, t.time_training, t.day_training, t.training, t.coacher_name FROM users u LEFT JOIN table_of_training t ON u.id = t.user_id`;
    mysql.query(query2, req.cookies?.UserID, (err, mysqlres1) => {
      if (err) {
        console.log(req.cookies?.UserID);
        console.log("error in getting all user " + err);
        res.status(400).send({message: "error in getting all user " + err});
        return;
      }
      else {
        console.log(mysqlres1);
        res.send({ trainings: mysqlres1 });
        return;
      }
    });
  }
};
```

כל משתמש יכול לכתוב ביקורת אך ורק לאימונים בהם נרשם אליהם.

אנו שלפנו את כל האימונים של אותו משתמש ושלחנו לקליינט במשתנים שונים.

:review -PUG

```
.container
form.form-container
  h1 Review class
  form(action='mailto:flow.studio@gmail.com' method='post' enctype='text/plain')
    select#training-selected(name='training')
      option(value='') Select a Training
    br
    label(for='score') The training score:
    select#score(name='score' size='1')
      option(value='1') 1
      option(value='2') 2
      option(value='3') 3
      option(value='4') 4
      option(value='5') 5
    p
      | A score of 5 indicates that the workout was challenging for you and your body.
    br
      | A score of 1 indicates that the workout was not challenging at all for you.
    .form-submit
      input#submit-btn(type='submit' value='Register')
```

ניתן לראות כי בצד לקוח אנו מקבלים את הנתונים ומאחדים אותם לשורה אחת כדי להציג למשתמש את האימונים שהוא יכול לכתוב עליהם ביקורת .

ניתן לראות את כל האימונים של הלקוח ברשימת גלילה נפתחת

לאחר הזנת הביקורת , הלקוח יפצל את הנתונים המאוחדים למפוצלים בשנית על מנת שנוכל לשלוח את זה לשרת ולהזין בבסיס נתונים הרלוונטי.

לאחר מכן אנו נמחק את הערכים הקיימים במשתנים על מנת שנוכל לשמור זאת למשתמש הבא שמתחבר.

```

// Function to populate the select element with available trainings
1 usage  A shahar bar
function populateSelect(trainings) {
    var selectElement = document.getElementById( "training-selected");

    // Clear the select element
    selectElement.innerHTML = "";

    // Add options to the select element for each training
    trainings.forEach(function (training) {
        var option = document.createElement( tagName: "option");
        option.value = training.time_training + " " + training.day_training + " " + training.Coacher_name + " " + training.training;
        option.textContent = training.time_training + " - " + training.day_training + " - " + training.Coacher_name + " - " + training.training;
        selectElement.appendChild(option);
    });
}

// Function to handle the form submission
1 usage  A shahar bar
function handleSubmit(event) {
    event.preventDefault();

    // Get the selected training value
    var selectElement = document.getElementById( "training-selected");
    var selectedTraining = selectElement.value;
    const score_select = document.querySelector( selectors: '#score');
    const score = score_select.options[score_select.selectedIndex].value;

    // Make an AJAX request to the server to register the training
    var xhr = new XMLHttpRequest();
    xhr.open( method: "POST", url: "/review_training", async: true);
    xhr.setRequestHeader( name: "Content-Type", value: "application/json");
    xhr.onreadystatechange = function () {
        if (xhr.readyState === 4 && xhr.status === 200) {
            var response = JSON.parse(xhr.responseText);
            console.log(response.message);
            // Handle success response

```

```

},
const values = selectedTraining.split(" ");
const time_training = values[0];
const day_training = values[1];
const Coacher_name = values[2];
const training_name = values[4];
// Prepare the data to send to the server
var data = {
    time_training: time_training,
    day_training: day_training,
    Coacher_name: Coacher_name,
    training: training_name,
    score: score
};
xhr.send(JSON.stringify(data))
.then((res) => res.json())
.then((data) => {
    // console.log("Response received from /my-profile");
    console.log(data);
    alert ("the user successfully wrote a review ");
    window.location.href = "my-profile";
    time_training.value = "";
    day_training.value = "";
    Coacher_name.value = "";
    training.value = "";
})
.catch((error) => {
    console.error("Error:", error);
});
}

// Make an AJAX request to the server to fetch the trainings
var xhr = new XMLHttpRequest();
xhr.open( method: "GET", url: "/mytrainings_review", async: true);

```


Review class

09:00:00 - Sunday - Matar Gersho ▾

The training score: 1 ▾

A score of 5 indicates that the workout was challenging for you and your body.
A score of 1 indicates that the workout was not challenging at all for you.

Register

Review class

09:00:00 - Sunday - Matar Gersho ▾

09:00:00 - Sunday - Matar Gershoni - Functional training
19:00:00 - Sunday - Katya Kalinoviz - Functional training
20:00:00 - Sunday - Amitay Gelre - Pilates Training
09:00:00 - Monday - Yoni cohen - TRX Training
10:00:00 - Monday - Matar Gershoni - Yoga Training
19:00:00 - Monday - Matar Gershoni - TRX Training
18:00:00 - Tuesday - Yafit halewer - Zumba Training
10:00:00 - Friday - Katya Kalinoviz - Yoga Training

Register

5.4 פונקציית הCRUD של review training גם מתקשרת לאופן זה היות והיא מעדכנת את הרשומה בה הלקוח רשום עם המאמן והאימון הספציפי הזה, ומעדכנת את הביקורת של הלקוח בטבלה זו .

```
Usage: shahar bar
const review_training = (req, res) => {
  const { time_training, day_training, Coacher_name, training, score } = req.body;
  const user_id = req.cookies?.UserID;
  // Add '%' to Coacher_name and training for partial matching
  const Coacher_name_like = '%' + Coacher_name + '%';
  const training_like = '%' + training + '%';
  const scoreValue = parseInt(score);

  const values = [scoreValue, time_training, day_training, Coacher_name_like, training_like, user_id];
  console.log(values);
  const updateQuery = `UPDATE table_of_training SET score = ? WHERE time_training = ? AND day_training = ? AND Coacher_name LIKE ? AND training LIKE ? AND user_id = ?`;
  mysql.query(updateQuery, values, (err, mysqlres) => {
    if (err) {
      console.log("error in updating registration: " + err);
      res.status(400).send({ message: "error in updating registration: " + err });
      return;
    } else {
      console.log(mysqlres);
      console.log("registration inserted successfully");
      res.send({ message: "Registration successful" });
      return;
    }
  });
};
```

ניתן לראות שנעשה שימוש בפונקציה זו גם כן בJS בצד הקליינט של review.

5.5 פונקציית get training וגם registertraining:

```
Usage: shahar bar
const getTrainings = (req, res) => {
  const query = `SELECT day_training, time_training, Coacher_name, training FROM table_of_training WHERE user_id=0 group by day_training, time_training, Coacher_name, training`;
  mysql.query(query, (err, mysqlres) => {
    if (err) {
      console.log("error in getting all the trainings: " + err);
      res.status(400).send({ message: "error in getting all trainings: " + err });
      return;
    } else {
      res.send({ trainings: mysqlres });
      return;
    }
  });
};
```

```
Usage: shahar bar
const registerTraining = (req, res) => {
  const { time_training, day_training, Coacher_name, training } = req.body;
  const user_id = req.cookies?.UserID;
  // Add '%' to Coacher_name and training for partial matching
  const Coacher_name_like = '%' + Coacher_name + '%';
  const training_like = '%' + training + '%';

  const values = [user_id, time_training, day_training, Coacher_name_like, training_like];
  console.log(values);
  const updateQuery = `UPDATE table_of_training SET user_id = ? WHERE time_training = ? AND day_training = ? AND Coacher_name LIKE ? AND training LIKE ?`;
  mysql.query(updateQuery, values, (err, mysqlres) => {
    if (err) {
      console.log("error in updating registration: " + err);
      res.status(400).send({ message: "error in updating registration: " + err });
      return;
    } else {
      console.log("registration inserted successfully");
      res.send({ message: "Registration successful" });
      return;
    }
  });
};
```

פונקציות אלה מציגות ללקוח אילו אימונים פנויים להרשמה ואילו לא.

כאשר אימון פנוי userid הוא כערך דיפולטיבי 0 ולכן כך בדקנו אילו אימונים פנויים.

בנוסף לכך, לאחר הצגת רשימת האימונים הזמינים להרשמה, היוזר יוכל להרשם לאימון בכך שנגדכן את הרשומה של אימון עם user id של המשתמש שהתחבר דרך cookie.

עמוד PUG מקושר : register for classes

```
.formbx
  .container
    form.form-container
      h1 Classes Registration
      p Attached all the optional classes for this week
      .triple-picture
        .bottomPicture
          img(src='/pics/yoga-class.jpeg' alt='Picture 1')
        .bottomPicture
          img(src='/pics/trx-class.jpeg' alt='Picture 2')
      br
      form.form-container(action='mailto:flow.studio@gmail.com' method='post' enctype='application/json')
        select#training-select(name='training')
          option(value='') Select a Training
        br
      .form-submit
        input#submit-btn(type='submit' value='Register')
```

Classes Registration

Attached all the optional classes for this week

09:00:00 - Sunday - Matar Gershoni - Functional training ▼

09:00:00 - Sunday - Matar Gershoni - Functional training ▲

10:00:00 - Sunday - Katya Kalinoviz - Pilates Training

11:00:00 - Sunday - Amitay Gelre - Strength Training

17:00:00 - Sunday - Yoni cohen - Yoga Training

18:00:00 - Sunday - Matar Gershoni - Zumba Training

21:00:00 - Sunday - Yafit halewer - Strength Training

11:00:00 - Monday - Katya Kalinoviz - Zumba Training

17:00:00 - Monday - Yafit halewer - Pilates Training

18:00:00 - Monday - Yoni cohen - Strength Training

20:00:00 - Monday - Katya Kalinoviz - Yoga Training

09:00:00 - Tuesday - Yafit halewer - Functional training

10:00:00 - Tuesday - Yoni cohen - Pilates Training

11:00:00 - Tuesday - Matar Gershoni - Strength Training

17:00:00 - Tuesday - Amitay Gelre - Yoga Training

18:00:00 - Tuesday - Yafit halewer - Zumba Training

19:00:00 - Tuesday - Yoni cohen - Functional training

20:00:00 - Tuesday - Matar Gershoni - Pilates Training

09:00:00 - Wednesday - Amitay Gelre - TRX Training

10:00:00 - Wednesday - Yafit halewer - Yoga Training

11:00:00 - Wednesday - Yoni cohen - Zumba Training ▼



Classes Registration

Attached all the optional classes for this week

09:00:00 - Sunday - Matar Gershoni - Functional training ▾

Register



אופן תצוגה זו דומה לאופן של review:

```
// Function to populate the select element with available trainings
// usage: shahar bar
function populateSelect(trainings) {
    var selectElement = document.getElementById( "training-select");

    // Clear the select element
    selectElement.innerHTML = "";

    // Add options to the select element for each training
    trainings.forEach(function (training) {
        var option = document.createElement( "option");
        option.value = training.time_training + " " + training.day_training + " " + training.Coacher_name + " " + training.training;
        option.textContent = training.time_training + " - " + training.day_training + " - " + training.Coacher_name + " - " + training.training;
        selectElement.appendChild(option);
    });
}

// Function to handle the form submission
// usage: shahar bar
function handleSubmit(event) {
    event.preventDefault();

    // Get the selected training value
    var selectElement = document.getElementById( "training-select");
    var selectedTraining = selectElement.value;

    // Make an AJAX request to the server to register the training
    var xhr = new XMLHttpRequest();
    xhr.open( "POST", url: "/registerTraining", async: true);
    xhr.setRequestHeader( "Content-Type", value: "application/json");
    xhr.onreadystatechange = function () {
        if (xhr.readyState === 4 && xhr.status === 200) {
            var response = JSON.parse(xhr.responseText);
            console.log(response.message);
            // Handle success response
            // Handle success response
        }
    }
}
```

```

xhr.setRequestHeader( name: "Content-Type", value: "application/json");
xhr.onreadystatechange = function () {
  if (xhr.readyState === 4 && xhr.status === 200) {
    var response = JSON.parse(xhr.responseText);
    console.log(response.message);
    // Handle success response
    // Handle success response
  }
};

const values = selectedTraining.split(" ");
const time_training = values[0];
const day_training = values[1];
const Coach_name = values[2];
const training_name = values[4];
// Prepare the data to send to the server
var data = {
  time_training: time_training,
  day_training: day_training,
  Coach_name: Coach_name,
  training: training_name
};
xhr.send(JSON.stringify(data))
.then((res) => res.json())
.then((data) => {
  // console.log("Response received from /my-profile");
  console.log(data);
  alert ("the user successfully register , please see all your trainings this week");
  window.location.href = "my-profile";
  time_training.value = "";
  day_training.value = "";
  Coach_name.value = "";
  training.value = "";
})
.catch((error) => {
  console.error("Error:", error);
});
}

```

```

65 }
66
67 // Make an AJAX request to the server to fetch the trainings
68 var xhr = new XMLHttpRequest();
69 xhr.open( method: "GET", url: "/getTrainings", async: true);
70 xhr.onreadystatechange = function () {
71   if (xhr.readyState === 4 && xhr.status === 200) {
72     var response = JSON.parse(xhr.responseText);
73     var trainings = response.trainings;
74     populateSelect(trainings);
75   }
76 };
77 xhr.send();
78
79 // Add event listener to the submit button
80 var submitBtn = document.getElementById( elementId: "submit-btn");
81 submitBtn.addEventListener( type: "click", handleSubmit);

```

ניתן לראות כי בצד לקוח אנו מקבלים את הנתונים של אילו אימונים פנויים להרשמה ומאחדים אותם לשורה אחת כדי להציג למשתמש את האימונים שהוא יכול להירשם אליהם..

ניתן לראות את כל האימונים של הלקוח ברשימת גלילה נפתחת .

לאחר הרשמה לאימון, הלקוח יפצל את הנתונים המאוחדים למפוצלים בשנית על מנת שנוכל לשלוח את זה לשרת ולהזין בבסיס נתונים הרלוונטי.

לאחר מכן אנו נמחק את הערכים הקיימים במשתנים על מנת שנוכל לשמור זאת למשתמש הבא שמתחבר.