313264111 315859660 205702665 308552249

אוניברסיטת בן-גוריון בנגב
Ben-Gurion University of the Negev

דו"ח פרויקט חלק ג'

פרח השכונות של ב"ש

קבוצה 7

מרצה: נעמה אילני צור

מתרגלת: לי ג'ולייט ימין

04.10.2021

# תוכן עניינים

הקדמה: ..... 3
שינויים מחלק ב': ..... 3
מבנה האתר והנחות: ..... 4
מבנה תיקיית הפרויקט: ..... 6
מימוש טפסים: ..... 7
מימוש פונקציה שמעבדת מידע מהלקוח ומחזירה לו תוצאה של עיבוד המידע: ..... 10
תוכן דינמי באתר ..... 10
חיבור לבסיס נתונים ושאילתות-SQL: ..... 11
נספח שאילתות נוספות בהן בוצע שימוש באתר ..... 13

313264111 315859660 205702665 308552249

## הקדמה:

בחרנו לפתח אתר עבור חנות הפרחים "פרח השכונות". מדובר באתר שישמש לסחר אלקטרוני עבור החנות וגם יאפשר יכולות נוספות, ביניהם: הרשמה לקורסי שזירה, אזור אישי המכיל את רכישותיו הקודמות, פרטי המשתמש, תאריכים מועדפים בהם ירצה לשלוח זר וכו'. בנוסף, קיים בסיס מידע דינמי ויעיל המאפשר החלפה ועדכון של מוצרים בכל רגע.

**שינויים מחלק ב':**

- ✓ יצירת סדר בתיקיות האתר סידור קבצי ה js , html, css כמתבקש בחלק זה
- ✓ יישום קבצי ejs המכילים את ה -header,footer meta על כלל עמודי האתר
- ✓ הוספת בדיקות תקינות לקלטי המשתמש
- ✓ יצירת הרשאות על בסיס הגדרת סטטוס המשתמש
- ✓ יצירת בסיס נתונים הן של מוצרי האתר והן של משתמשי המערכת
- ✓ מימוש פונקציות המעבדות מידע מן הלקוח ומחזירות לו תשובה בהתאם למידע הנמצא בסיס הנתונים
- ✓ הוספת תוכן דינמי לאתר

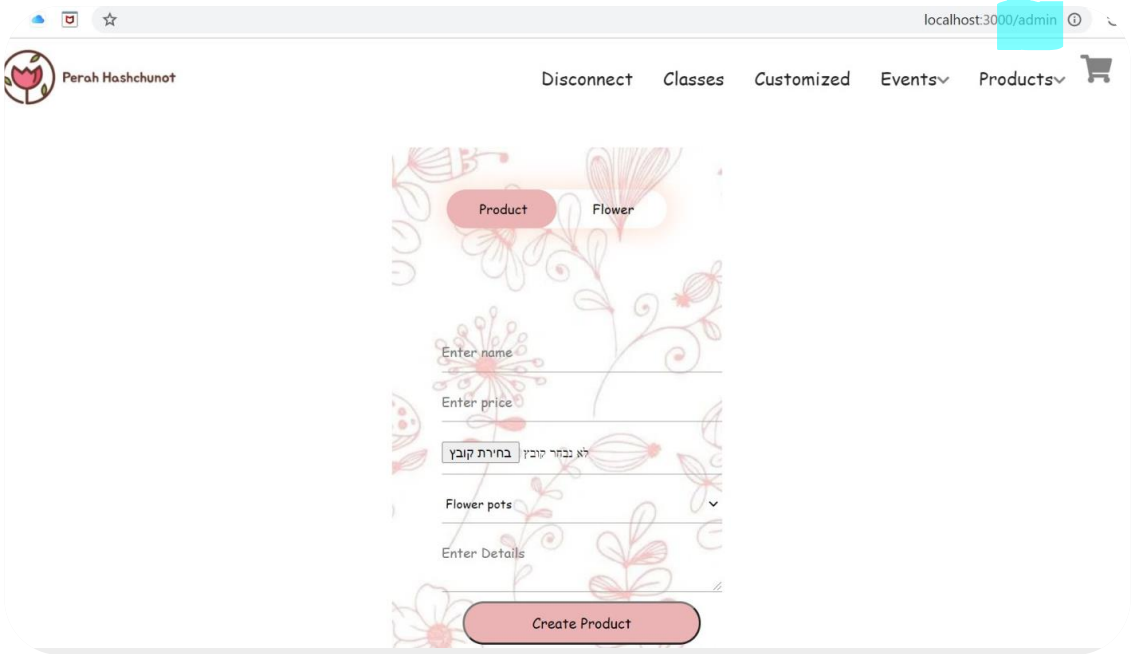
**מבנה האתר והנחות:**

עמוד הרשמה/התחברות לאתר : משתמש יכול לבצע רישום או כניסה לאתר בעמוד זה. במידה והוא משתמש רשום באפשרותו להתחבר לאתר ע"י הזנת סיסמה ומייל תואמים- אם יזין פרטים שאינם תואמים למסד הנתונים תוצג לו הודעת שגיאה. במידה והמשתמש לא רשום, באפשרותו להירשם לאתר ולהתחבר.

Remember me- בזמן ביצוע כניסת משתמש לאתר ישנה אפשרות לבחור ב "remember me" כך שהמשתמש לא יצטרך להתחבר בכל פעם מחדש.

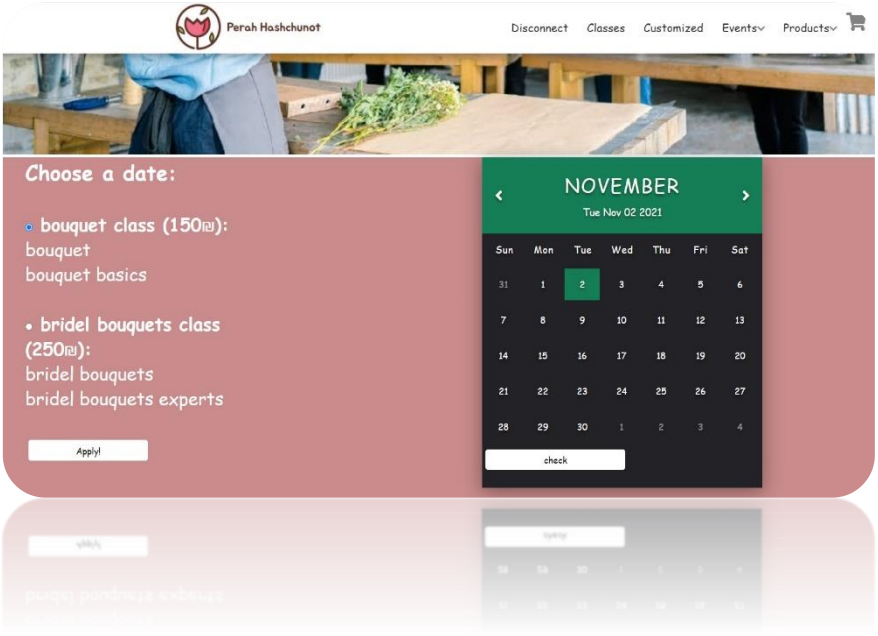
הרשאות : בעת הרשמת משתמש הוא מוגדר מתוך ברירת מחדל כ"לקוח" , קיימת אפשרות לערוך הגדרה זו ולהגדיר משתמשים רשומים בלבד כמנהלים.

**לקוח** בעל הרשאות לבצוע רישום/כניסה לאתר, בחירת מוצרים והוספתם לסל, הרכבת זר אישי, רישום לקורסים ע"פ זמינות. בנוסף, יכול המשתמש לערוך את עגלת הקניות שלו, לצפות ולערוך את פרטיו באזור אישי (מפורט בהמשך). משתמש המוגדר כ**מנהל** רשום יוכל לבצע שינויים במאגר הנתונים- ניהול מוצרים וקורסים : הוספה, מחיקה ועריכה של מוצרים וקרסים. מצורף צילום מסך הממחיש את יכולת המנהל להוסיף מוצר ע"י הגדרת פרטיו והוספת קובץ תמונה.




אזור אישי-דף המכיל את פרטיו האישיים של המשתמש המסונכרנים עם פרטי המשתמש אשר הוזנו בזמן ההרשמה וניתנים לעריכה ע"י המשתמש. בנוסף, קיימות אפשרויות לבחירת תמונה לפי העדפותיו מתוך אפשרויות קיימות, הזנת תאריכים אשר מעוניין לשמור באתר על מנת שבזמן תאריכים אלו וצפייה בהיסטורית הזמנות אשר בוצעו ע"י המשתמש.

קורסים- יכולים להתווסף ע״י מנהלים רשומים בלבד. קיימת הגבלת משתתפים עבור כל קורס המוגדרת בעת הוספת הקורס למאגר. במידה ומשתמש ירצה להירשם לקורס שנמצא בתפוסה מלאה- הוא יקבל הודעת שגיאה. הקורסים מוצעים ע״פ תאריך- כלומר, בעת פתיחת עמוד הקורסים יבחר המשתמש תאריך ובעת לחיצה על כפתור ״**check**״ יוצגו לו הקורסים אשר מתקיימים בתאריך הנבחר. המשתמש יוכל לבצע הרשמה לקורס בעת לחיצה על כפתור ״**apply**״. מצ״ב תמונה המתארת את הליך הרשמה לקורס.




קולקציית מוצרים-באתר קיימים דפי קולקציות הניתנים לניהול ושינוי ע״י מנהל, עבור המוצרים השונים. ניתן להגיע אליהם בעזרת סרגל הניווט, לסנן את תוצאות המוצרים המוצגים ע״פ מחיר.

הרכבת זר- קיימת אפשרות להרכיב זר באופן עצמאי באתר. הזר מורכב מתוך מאגר פרחים המוצג ללקוח, אשר מגדיר את הפרחים שברצונו להוסיף ואת גודל הזר רצוי. קולקציית הפרחים דינמית ומשתנה בהתאם להגדרות המנהל.

דפי מוצר- לכל מוצר באתר קיים דף המתאר את כלל פרטיו ובו מוצגת תמונת המוצר וכמות אותה הלקוח רוצה לרכוש שבאפשרותו לערוך, קיימת אפשרות הוספה לסל ומעבר לסיום רכישה.

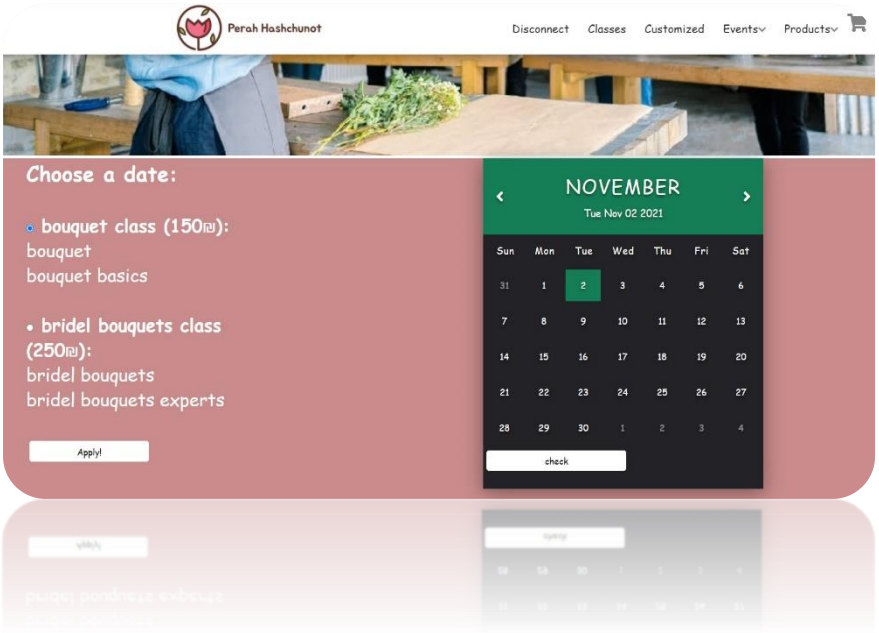
**מבנה תיקיית הפרויקט:**

- **Assats**
  - Css- home, payment עיצוב האתר מוגדר ע״י שני קבצים אלו, אשר בהם מוגדרת בין היתר תצוגה המשתנה בהתאם למסך המשתמש
  - Images – מאגר התמונות באתר
  - Js- index, scheduler, slider קבצי הJS המגדירים פונקציות שונות באתר ביניהם פונ׳ לוח שנה
- **Node modules**
- **Server**
  - Controllers -מכיל את השאילתות והקשר לSQL (ראי פירוט בהמשך).
  - Router.js – מאפשר קישוריות לפונקציות שונות בדפי האתר, כך שהקריאה לפונקציות מתבצעת בצורה מודולרית ויעילה. ניתן לראות בתחילת העמוד את הקישורים השונים בעמודים השונים לכלל הפונקציות שמופיעות בפרויקט.
  - Services.js – מתן שירות לדפים המופנים ע״י ה Router ומקנים להם את הקישוריות לפעולות השונות שמתבצעות בפרויקט וביניהם קבצי ה-Controllers
- **Views**
  - ejs - בחרנו להשתמש ב- ejs המאפשר שימוש במודולריות שכן ניתן לראות כי ה Footer וה- Header נכתבו פעם אחת בלבד והופיעו בכל דפי האתר באמצעות קריאה לדפים אלו. בנוסף, שימוש ב-ejs מאפשר קישוריות יעילה ומהירה לבסיס הנתונים.
  - Footer, Header ו-Meta – כאמור, דפי HTML אשר השימוש בהם מתבצע ע״י קריאה מתוך- ejs, קבצי- sql המכילים את מאגר הDATA באתר.
- **Config.env** -מכיל את פרטי החיבור לבסיס הנתונים
- **Json**
- **Server.js** – דף המקשר את דפי הפרויקט ל port ולבסיס הנתונים באמצעות קריאה לconfig ומאפשר את השימוש בקבצי ה-ejs .

## מימוש טפסים:

במהלך הפרויקט מתרחשים מימושי טפסים רבים. מצ״ב שתי דוגמאות למימוש טפסים:

הרשמה לקורס שזירה מתבצעת על ידי בחירת המשתמש באחד מתאריכי החודש,בהם מוצאים קורסים. בעת ההרשמה נבדקת ההתניה האם יש מספיק מקום בקורס ובמקרה שאין תצא הודעה לבחירת תאריך אחר, אם יש מקום בתאריך הרצוי הלקוח יועבר לעמוד מילוי פרטים לתשלום עבור ההזמנה, לאחר מכן יישלח הלקוח לעמוד היסטוריית הרכישות שלו שם יוכל לעיין בהזמנה הנוכחית וברכישות הקודמות שערך.

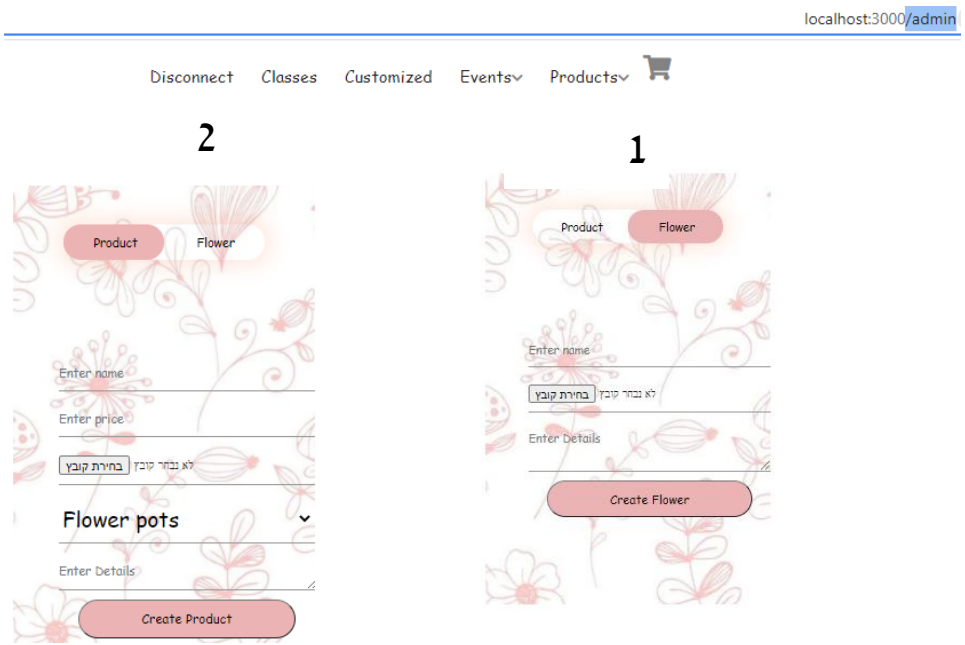


מימוש הטופס

```
      <form action="/classes">
        <input type="date" name="date" id="form-dateT" hidden required>
        <button class="button" style="vertical-
align:middle"><span>check</span></button>

      </form>
</div>

    <form action="/classes/join">
      <input type="date" name="date" id="form-date" hidden required>
      <div class="scheduleh1"><h3>Choose a date:</h3>

        <br>
        <%for(let i = 0; i< classes.length; i++){%>
        <h4><input type="radio" name="id" value="<%=classes[i].id%>" <%
if(i==0){%>checked<%}%> /> <%=classes[i].name%> (<%=classes[i].price%>₪
):</h4>
        <p><%=classes[i].lecturer%></p>
        <%=classes[i].description%> <br><br>
        <%}%>
        <button class="button" style="vertical-
align:middle"><span>Apply!</span></button>
      </div>
    </form>
```

**<u>טופס הוספת מוצר למערכת:</u>**

באפשרות המשתמש המוגדר כמנהל, להוסיף מוצרים שונים באמצעות מילוי הטפסים בעמוד ה-admin , על המנהל לבחור את אחת משתי הקטגוריות המוצעות לעיל, לסווג את המוצר, להוסיף תמונה , לקבוע מחיר ושם עבור המוצר אותו רוצה להוסיף, ההוספה מתבצעת בעמוד שמוצג להלן:



מימוש הטופס לשתי הקטגוריות flower ו- products:

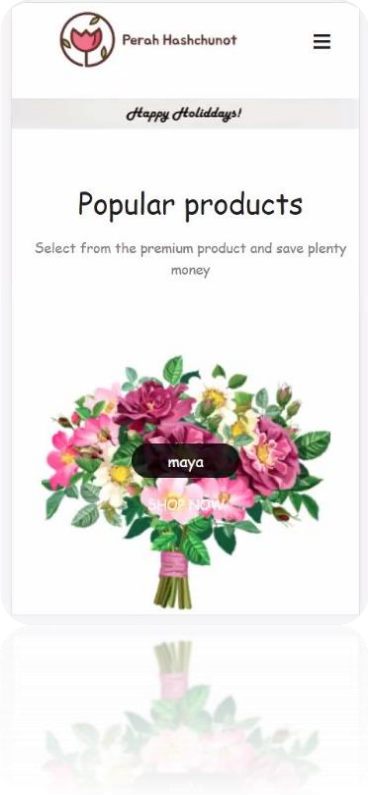
```
            <form  id="login" action="/admin/product" method="POST" enc
type="multipart/form-data" class="inputDit">

                <input type="text" class="input-
field" placeholder="Enter name" name="name" required pattern="[ A-Za-
z]{1,50}" title="Enter valid name">
                <input type="number" class="input-
field" placeholder="Enter price" name="price" required>
                <input type="file" class="input-
field" placeholder="Enter picture" name="picture" required>
                <select class="input-
field" placeholder="Enter type" name="type" required>
                    <option>Flower pots</option>
                    <option>Bouquets</option>
                    <option>Flower crowns</option>

                    <option>Table arrangements</option>
                    <option>Bridal bouquet</option>
                </select>
                <textarea class="input-
field" placeholder="Enter Details" name="description" required></textar
ea>
                <button type="submit" class="submit-
button" style="vertical-
align:middle"><span>Create Product</span></button>

            </form>

            <form  id="register" action="/admin/flower" enctype="multip
art/form-data" method="POST" class="inputDit">
                <input type="text" class="input-
field" placeholder="Enter name" name="name" required pattern="[ A-Za-
z]{1,50}" title="Enter valid name">
                <input type="file" class="input-
field" placeholder="Enter picture" name="picture" required>
                <textarea class="input-
field" placeholder="Enter Details" name="description" required></textar
ea>
                <button type="submit" class="submit-
button" style="vertical-
align:middle"><span>Create Flower</span></button>

            </form>
```

## מימוש פונקציה שמעבדת מידע מהלקוח ומחזירה לו תוצאה של עיבוד המידע:

בעת רישום לקוח לקורס, בוחר הלקוח תאריך בו מעוניין להירשם לקורס והמערכת מחזירה לו רשימת קורסים מהם יכול לבחור את הקורס הרלוונטי לו. לאחר מכן, המערכת מקבלת כקלט את הקורס אשר הלקוח מעוניין להירשם אליו לאחר מכן מתבצעת בדיקה במערכת, בנוגע לתפוסת הקורס- כלומר האם נשאר מקום פנוי בקורס בתאריך המבוקש- במידה והקורס מלא תשלח הודעת שגיאה עם המלצה לבחירת תאריך חדש, אחרת- ההרשמה תבוצע בהצלחה והלקוח יועבר לדף החיוב.

**<u>התאמת תצוגת האתר:</u>**

תצוגת האתר דינמית ותומכת במכשירים ומסכים בגדלים שונים. לדוגמה שימוש באתר מתוך מכשיר MOBILE GALAXY:




## תוכן דינמי באתר

באתר קיים מאגר דינמי של ששת המוצרים הפופולאריים ביותר. המאגר מתעדכן בהתאם לקניות המתבצעות באופן שוטף באתר, כך שבכל רגע נתון יוצגו ששת המוצרים הנמכרים ביותר נכון לאותו הזמן. מצ"ב השאילתה להמחשה:

```
exports.GetPopularProducts = async () => {
        let conn = await getConnection();
        query = `SELECT *,
                        (SELECT count(*)
                         FROM orderitem
                         WHERE item_id=product.id AND item_type='product') as popularity
                         FROM product
                         ORDER BY popularity DESC
                         LIMIT 0,6`;
```

313264111 315859660 205702665 308552249

# חיבור לבסיס נתונים ושאילתות -SQL:

רשימת טבלאות בבסיס הנתונים:

| | שם השדה | טיפוס | תיאור |
|---|---|---|---|
| **cart** | **id** | int AI PK | מספר סידורי עבור עגלת הלקוח. |
| | user_id | int | מספר המזהה של הלקוח, מספרים בלבד. |
| | product_type | varchar(100) | מייצג את סוג המוצר. |
| | product_id | int | מספר סידורי ייחודי לכל למוצר,מספרים בלבד. |
| | quantity | int | כמות היחידות שיבחר הלקוח לקנות ממוצר אחד,מספרים בלבד |
| **class** | **id** | int AI PK | מזהה הקורס |
| | max_participants | int | מספר המתאר את מגבלת התלמידים שיוכלים להגיע לשיעור מס |
| | description | text | שדה המתאר את סוג הקורס. |
| | name | varchar(100) | שם הקורס משתנה לפי סוגי ההקורסים השונים . |
| | lecturer | varchar(100) | מהות ההרצאה |
| | date | datetime | מועד הקורס |
| | price | float | מחיר הקורס משתנה לפי סוג הקורס. |
| **class join** | **id** | int AI PK | מספר סידורי עבור הקורס שבחר |
| | **user_id** | int | מספר זהות הלקוח. |
| | **date** | date | מסמל את התאריך שנבחר, במסגרת ימים ,חודשים, ושנים. |
| **contact_us** | **id** | int AI PK | מזהה הפניה |
| | name | varchar(100) | שם המשתמש |
| | phone | varchar(100) | טלפון לחזרה |
| | email | varchar(100) | מייל הפונה |
| | message | text | מהות הפניה |
| **credit_card** | **id** | int AI PK | מספר סידורי עבור פרטי כרטיס האשראי. |
| | order_id | int | מזהה הזמנה |
| | type | int | חברת הכרטיס |
| | number | varchar(30) | מספר כרטיס הלקוח. |
| | expiry_year | int | תאריך תפוגה שנים,מספרים בלבד. |
| | expiry_month | int | תאריך תפוגה חודשים,מספרים בלבד. |
| | cvv | int | מספר מזהה בגב הכרטיס,מספרים בלבד. |
| | first_name | varchar(100) | שם הלקוח . |
| | last_name | varchar(100) | שם משפחה של הלקוח. |
| **customize_product** | **id** | int AI PK | מזהה סל |
| | size | varchar(11) | גודל המוצר אותו בוחר הלקוח (S,M,L) |
| | price | float | מחיר המוצר. |
| **customize_product_items** | **id** | int AI PK | מזהה בחירה |
| | customize_id | int | מזהה לקוח |
| | flower_id | int | מזהה פרח |
| **flower** | **id** | int AI PK | מזהה הפרח |
| | name | varchar(100) | שם הפרח. |
| | description | text | תיאור על הפרח. |
| | picture | varchar(100) | נתיב התמונה של הפרח. |
| **orderitem** | **id** | int AI PK | מזהה עגלה |
| | order_id | int | מזהה הזמנה |
| | item_id | int | מזהה מוצר |
| | item_type | varchar(20) | סוג מוצר - תיאור |
| **orders** | **id** | int AI PK | מזהה הזמנה |
| | date | datetime | תאריך ההזמנה. |
| | delivery_method | varchar(100) | סוג משלוח |
| | delivery_price | float | דמי משלוח |
| | recipient_name | varchar(100) | שם בעל הכרטיס |
| | recipient_phone | varchar(100) | מספר טלפון לשליחת קבלה |
| | recipient_addres | varchar(256) | כתובת לשליחת קבלה |
| **users** | **id** | int AI PK | מזהה המשתמש- מורכב ממספר מזהה בעל 9 ספרות |
| | first_name | varchar(100) | שם פרטי- אותיות באנגלית בלבד |
| | last_name | varchar(100) | שם משפחה- אותיות באנגלית בלבד |
| | email | varchar(100) | אימייל- בדיקת תקינות כי נכתב בפורמט המתאים |
| | password | varchar(100) | סיסמא הנקבעת עפ בחירת המשתמש |
| | address_city | varchar(100) | כתובת המשתמש אותיות בלבד |
| | address_street | varchar(100) | כתובת המשתמש אותיות בלבד |
| | address_zipcode | int | מיקוד- מספר בלבד |
| | address_number | varchar(100) | מספר/כניסה לבנין |
| | date_of_birth | date | תאריך יום הולדת |
| | picture | varchar(100) | תמונה מתוך מאגר תמונות |
| | role | varchar(20) | תפקיד המשתמש- לקוח/מנהל |
| **product** | **id** | int AI PK | מזהה המוצר |
| | name | varchar(100) | שם המוצר. |
| | price | float | מחיר המוצר. |
| | picture | varchar(100) | נתיב תמונת המוצר. |
| | type | varchar(100) | סוג המוצר. |
| | description | text | תיאור המוצר. |
| **special_dates** | **id** | int AI PK | מספר סידורי, אשר מקנה מזהה עבור כל תאריך המשוייך למשת |
| | user_id | int | מזהה המשתמש |
| | date | date | תאריך שאותו הלקוח רוצה לשמור |
| | title | varchar(100) | הכותבת המתארת מה המאורע המתרחש בתאריך |

**שאילתות**

מודגשות בכחול ארבעת השאילתות אשר עונות על ההנחיות – הוספה, עדכון מחיקה ובחירה. שאר השאילתות מפורטות בהמשך.

שאילתת הוספת משתמש חדש, מתבצעת בעת הרשמה משתמש לאתר:

```
exports.RegisterUser = async (fname, lname, email, password) => {
        let conn = await getConnection();
        let user = await GetUserByEmail(email);
        if(user.length!==0)
                return false;
        let query = `INSERT INTO
                        users(first_name, last_name, email, password, role, picture)
                        VALUES(?,?,?,?,?,?)`;
        let data = [fname,lname,email,password,"customer","user.jpg"];
        return new Promise(function(resolve, reject){
                conn.query(query,data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(true);
                });
        });
}
```

שאילתת עדכון פרטי משתמש :

```
exports.UpdateUser=async(field_name, field_value, id, email)=>{
        let conn = await getConnection();
        let query = `UPDATE users SET
                                ${field_name}=? WHERE id=?`;
        let data = [field_value, id];
        console.log(query, data);
        return new Promise(function(resolve, reject){
                conn.query(query,data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(GetUserByEmail(email));
                });
        });
};
}
```

בחירת שיעור מתוך קורסים :

```
exports.JoinClass = async(u, d, c)=>{
        let conn = await getConnection();
        let query = "INSERT INTO class_join(user_id, date, class_id) VALUES(?,?,?)"
        let data = [u,d,c];
        return new Promise(function(resolve, reject){
                conn.query(query, data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
```

```
                });
        });
```

מחיקת מוצר מתוך עגלת קניות:

```
query = "DELETE FROM cart WHERE user_id=? AND product_id=? AND product_type=?";
        let _ = await new Promise(function(resolve, reject){
                conn.query(query, [data[0].user_id, data[0].product_id, data[0].product_type], function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
                });
        });
        return CreateCart(data[0].user_id, data[0].product_type, data[0].product_id, quantity)
}
```

# נספח שאילתות נוספות בהן בוצע שימוש באתר

***שאילתות ניהול USER:***

*שאילתה שמחזירה את מזהה המשתמש לפי מייל:*

```
let GetUserByEmail = exports.GetUserByEmail = async(email)=>{
        let conn = await getConnection();
        let query = `SELECT * FROM users WHERE email=?`;
        let data = [email];
        return new Promise(function(resolve, reject){
                conn.query(query,data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
                });
        });
}
```

שאילתת התחברות המשתמש- בדיקת התאמה בין אימייל לסיסמה:

```
exports.LoginUser = async(email, password)=>{
        let conn = await getConnection();
        let query = `SELECT * FROM users WHERE email=? AND password=?`;
        let data = [email, password];
        return new Promise(function(resolve, reject){
                conn.query(query,data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
                });
        });
}
```

שאילתות ניהול עגלות קניות והזמנות:

יצירת עגלת קניות חדשה

```
let CreateCart = exports.CreateCart = async(user_id, product_type, product_id, quantity) => {
        let conn = await getConnection();
```

```
	let query = `INSERT INTO cart(user_id, product_type, product_id, quantity)
		VALUES(?,?,?,?);`;
	let data = [user_id, product_type, product_id, quantity];
	return new Promise(function(resolve, reject){
		conn.query(query, data, function(err, result){
			if(err)
				reject(err);
			resolve(result);
		});
	});
```

}

שליפת מידע מתוך עגלת קניות

```
exports.GetCartData = async(user_id)=>{
	let conn = await getConnection();
	query = `SELECT *, cart.id as cart_id, sum(cart.quantity) as total_count FROM cart, product WHERE cart.user_id=? AND cart.product_id=product.id AND cart.product_type='product' GROUP BY cart.product_id`
	let data_1 = await new Promise(function(resolve, reject){
		conn.query(query, [user_id], function(err, result, fields){
			if(err)
				reject(err);
			resolve(result);
		});
	});
	query = `SELECT *, cart.id as cart_id, sum(cart.quantity) as total_count FROM customize_product, cart WHERE cart.user_id=? AND cart.product_id=customize_product.id AND cart.product_type='customize' GROUP BY cart.product_id`
	let data_2 = await new Promise(function(resolve, reject){
		conn.query(query, [user_id], function(err, result, fields){
			if(err)
				reject(err);
			resolve(result);
		});
	});
	for(let i = 0; i<data_2.length; i++){
		query = "SELECT flower.* FROM flower, customize_product_items WHERE customize_product_items.flower_id=flower.id AND customize_product_items.customize_id=?"
		data_2[i]["flowers"] = await new Promise(function(resolve, reject){
			conn.query(query, [data_2[i].product_id], function(err, result){
				if(err)
					reject(err);
				resolve(result);
			});
		});
	}
	return [data_1, data_2];
}
```

עדכון עגלת קניות

```
exports.UpdateCart = async (cart_id, quantity) =>{
	let conn = await getConnection();
	let query = "SELECT * FROM cart WHERE id=?";
	let data = await new Promise(function(resolve, reject){
		conn.query(query, [cart_id], function(err, result){
			if(err)
```

```
                                        reject(err);
                                resolve(result);
                        });
                });
                query = "DELETE FROM cart WHERE user_id=? AND product_id=? AND product_type=?";
                let _ = await new Promise(function(resolve, reject){
                        conn.query(query, [data[0].user_id, data[0].product_id, data[0].product_type], function(err, result){
                                if(err)
                                        reject(err);
                                resolve(result);
                        });
                });
                return CreateCart(data[0].user_id, data[0].product_type, data[0].product_id, quantity)
}
```

## החזרת היסטורית הזמנות עבור משתמש :

```
exports.ViewHistory = async(user_id) =>{
                let orders = await ViewProductHistory(user_id);
                let classes = await ViewClassHistory(user_id);
                return {orders:orders, classes:classes}
}
```

## שחזור הזמנות קודמות

```
et ViewProductHistory = exports.ViewProductHistory = async(user_id)=>{
                let conn = await getConnection();
                query = "SELECT * FROM orders WHERE user_id=?"
                let orders = await new Promise(function(resolve, reject){
                        conn.query(query, [user_id], function(err, result){
                                if(err)
                                        reject(err);
                                resolve(result);
                        });
                });
                for(let i = 0; i<orders.length; i++){
                        query = "SELECT * FROM product, orderitem WHERE orderitem.order_id=? AND
orderitem.item_type='product' AND orderitem.item_id=product.id GROUP BY product.id"
                        orders[i]["products"] = await new Promise(function(resolve, reject){
                                conn.query(query, [orders[i].id], function(err, result){
                                        if(err)
                                                reject(err);
                                        resolve(result);
                                });
                        });
                        query = "SELECT * FROM customize_product, orderitem WHERE orderitem.order_id=? AND
orderitem.item_type='customize' AND orderitem.item_id=customize_product.id GROUP BY customize_product.id"
                        orders[i]["customize"] = await new Promise(function(resolve, reject){
                                conn.query(query, [orders[i].id], function(err, result){
                                        if(err)
                                                reject(err);
                                        resolve(result);
                                });
                        });
                }
                return orders;
```

```
}

et ViewClassHistory = exports.ViewClassHistory = async(user_id)=>{
	let conn = await getConnection();
	let query = "SELECT class_join.*, class.name as name, class.price as price FROM class_join, credit_card, class WHERE
credit_card.order_id=class_join.id AND credit_card.type=1 AND class_join.user_id=? AND class.id=class_join.class_id"
	return new Promise(function(resolve, reject){
		conn.query(query, [user_id], function(err, result){
			if(err)
				reject(err);
			resolve(result);
		});
	});
};
```

שמירת כרטיסי אשראי

```
exports.SaveCard = async(data)=>{
	let conn = await getConnection();
	let query = `INSERT INTO credit_card(order_id, type, number, expiry_year,expiry_month, cvv, first_name, last_name)
			VALUES(?,?,?,?,?,?,?,?);`;
	return new Promise(function(resolve, reject){
		conn.query(query, data, function(err, result){
			if(err)
				reject(err);
			resolve(result);
		});
	});

}
```

Class שאילתות שיעורי שזירה :

יצירת שיעור חדש :

```
 exports.CreateClass = async(name, lecturer, description, max, price, Date)=>{
	let conn = await getConnection();
	let query = `INSERT INTO class(name,lecturer,description, price, max_participants, Date) VALUES(?,?,?,?,?,?)`;
	let data = [name,lecturer, description, price, max, Date];
	return new Promise(function(resolve, reject){
		conn.query(query,data, function(err, result){
			if(err)
				reject(err);
			resolve(result);
		});
	});
}
```

צפייה בפרטי שיעור :

```
exports.ViewClasses = async(d)=>{
	let conn = await getConnection();
	let query = "SELECT * FROM class where date=?";
	let data = [d];
	console.log(d);
	return new Promise(function(resolve, reject){
		conn.query(query, data, function(err, result){
```

```
                    if(err)
                        reject(err);
                    resolve(result);
                });
        });
}
```

הרשמה לשיעור :

```
exports.JoinClass = async(u, d, c)=>{
        let conn = await getConnection();
        let query = "INSERT INTO class_join(user_id, date, class_id) VALUES(?,?,?)"
        let data = [u,d,c];
        return new Promise(function(resolve, reject){
                conn.query(query, data, function(err, result){
                    if(err)
                        reject(err);
                    resolve(result);
                });
        });
```

}

בדיקת מכסת משתתפים- בעת הרשמת לקוח לקורס מתבצעת בדיקת סטטוס כמות נרשמים, במידה והקורס מלא, כלומר מספר המשתתפים הינו מקסימלי המותר לקורס ההרשמה נדחת, אחרת מאושרת

```
exports.AvailableClass = async(id, d) =>{
        let conn = await getConnection();
        let query = "select (select count(*) as participants from class_join where class_id=?) as participants, max_participants from class where class.id=?";
        let data = [id,id];
        return new Promise(function(resolve, reject){
                conn.query(query, data, function(err, result){
                    if(err)
                        reject(err);
                    resolve(result);
                });
        });
}
```

פעולות יצירת קשר

```
exports.ContactUs = async (name, email, phone, message) => {
        let conn = await getConnection();
        let query = `INSERT INTO
                    contact_us(name, email, phone, message)
                    VALUES(?,?,?,?)`;
        let data = [name,email,phone,message];
        return new Promise(function(resolve, reject){
                conn.query(query,data, function(err, result){
                    if(err)
                        reject(err);
                    resolve(result);
                });
        });
```

}

ניהול פרחים- פרחים בודדים לצורך הרכבת זר

הוספת פרח חדש למאגר(ע"י מנהל בלבד) :

```
exports.CreateFlower = async(name, picture, description) => {
	let conn = await getConnection();
	let query = `INSERT INTO flower(name, picture, description)
				VALUES(?,?,?);`;
	let data = [name, picture, description];
	return new Promise(function(resolve, reject){
		conn.query(query, data, function(err, result){
			if(err)
				reject(err);
			resolve(result);
		});
	});
```

צפייה בקולקציית הפרחים :

```
exports.GetAllFlowers = async()=>{
	let conn = await getConnection();
	let query = "SELECT * FROM flower";
	return new Promise(function(resolve, reject){
		conn.query(query, function(err, result){
			if(err)
				reject(err);
			resolve(result);
		});
	});
}
```

יצירת זר חדש ע״י הרכבה עצמית :

```
exports.CreateBouqeut = async (user_id, list_of_items, price_type) => {
	let conn = await getConnection();
	let price = 150;
	if(price_type=="small")
		price = 80;
	else if(price_type=="medium")
		price=120;
	else
		price=150;
	let query = "INSERT INTO customize_product(size, price) VALUES(?,?)";
	let data = [price_type, price];
	let res = await new Promise(function(resolve, reject){
		conn.query(query,data, function(err, result){
			if(err)
				reject(err);
			resolve(result);
		});
	});
	let ID = res.insertId;
	for(let i = 0; i<list_of_items.length; i++) {
		query = "INSERT INTO customize_product_items(customize_id, flower_id) VALUES(?,?)"
		data = [ID, list_of_items[i]];
		let res = await new Promise(function(resolve, reject){
			conn.query(query,data, function(err, result){
				if(err)
```

```
                                        reject(err);
                                resolve(result);
                        });
                });
        }
        res = await CreateCart(user_id, "customize", ID, 1);
        return res;
}
```

פעולות על מוצרים

הוספת מוצר חדש:

```
exports.CreateProduct = async(name, price, picture, type, description) => {
        let conn = await getConnection();
        let query = `INSERT INTO product(name, price, picture, type, description)
                        VALUES(?,?,?,?,?);`;
        let data = [name, price, picture, type, description];
        return new Promise(function(resolve, reject){
                conn.query(query, data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
                });
        });
}
```

צפייה במוצר:

```
exports.GetSingleProduct = async(ID)=>{
        let conn = await getConnection();
        query = `SELECT * FROM product WHERE id=?`;
        return new Promise(function(resolve, reject){
                conn.query(query,[ID], function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
                });
        });
};
```

צפייה במוצרים לפי סיווג מחיר:

```
exports.GetProducts = async(type, rangeL=0, rangeR=150, page=1)=>{
        let conn = await getConnection();
        query = `SELECT * FROM product WHERE type=? AND price BETWEEN ? AND ? LIMIT ?,8`;
        let data =[type, rangeL, rangeR, (page-1)*8];
        return new Promise(function(resolve, reject){
                conn.query(query,data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
                });
        });
};

exports.GetCountOfProducts = async(type, rangeL, rangeR)=>{
        let conn = await getConnection();
```

```
        query = `SELECT count(*) as total FROM product WHERE type=? AND price BETWEEN ? AND ?`;
        let data =[type, rangeL, rangeR];
        return new Promise(function(resolve, reject){
                conn.query(query,data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(result[0].total);
                });
        });
}
```

צפייה בעשרת המוצרים הנמכרים ביותר

```
exports.GetPopularProducts = async () => {
        let conn = await getConnection();
        query = `SELECT *,
                        (SELECT count(*)
                         FROM orderitem
                         WHERE item_id=product.id AND item_type='product') as popularity
                         FROM product
                         ORDER BY popularity
                         LIMIT 0,10`;

        return new Promise(function(resolve, reject){
                conn.query(query, function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
                });
        });
}
```

ניהול תאריכים מיוחדים :

יצירת "תאריך מיוחד" :

```
exports.CreateSpecialDate = async(user_id, date, title) => {
        let conn = await getConnection();
        let query = `INSERT INTO special_dates(user_id, date, title)
                        VALUES(?,?,?);`;
        let data = [user_id, date, title];
        return new Promise(function(resolve, reject){
                conn.query(query, data, function(err, result){
                        if(err)
                                reject(err);
                        resolve(result);
                });
        });
```

הצגת תאריך לידת המשתמש בצורה אוטומטית כ"תאריך מיוחד"

```
exports.GetSpecialDates = async(user_id)=>{
        let conn = await getConnection();
        let query = `SELECT * FROM special_dates WHERE user_id=?`;
        let data = [user_id];
        return new Promise(function(resolve, reject){
```

```
        conn.query(query, data, function(err, result){
            if(err)
                reject(err);
            resolve(result);
        });
    });
```