פרויקט הקורס

בניית מערכות ממוחשבות מבוססות אינטרנט

חלק 3- צד שרת

מרצה:

נעמה אילני צור

MY PASTRY

מגישה: שלי כהן

תעודת זהות: 208308122

1. מבנה תיקיית הפרויקט

תיקיית **static** שכוללת בתוכה את כל הקבצים הסטטיים בפרויקט: בתוכה ניתן למצוא את תיקיית **media** שכוללת בתוכה את כל התמונות שבהן השתמשתי בבניית האתר, מחלק א' ועד הסוף. תיקיית **css** המכילה קובץ אשר קובע את העיצוב של כל העמודים באתר . תיקיית **js** שכוללת בתוכה את קבצי הגאווה סקירפט בצד הלקוח מחלק ב' עם שינויים שבוצעו בעקבות חלק ג' של הפרויקט.

תיקייה נוספת היא תיקיית views הכוללת בתוכה את כל קבצי ה pug של הפרויקט שנוצרו .בנוסף, קיימות תיקיית **db** הכוללת בתוכה קבצים אשר קשורים ל DB ותיקיית **node_models** שנותרה ללא שינוי.

2. טיפול בבקשות לקוח

בקשת הלקוח הנוצרת היא בחירה ורכישה של מאפים ומהתפריט אותו ניתן לראות באתר. הבחירה מתבצעת על ידי כפתור add to cart הנמצא ליד כל פריט. כך הלקוח יכול לבחור את כל הפריטים אשר ברצונו לרכוש ולאחר מכן על ידי כפתור buy now אשר מופיע רק כאשר קיים לפחות פריט אחד בעגלה הוא יכול לעבור לדף הרכישה בו עליו למלא את פרטיו. כמו כן, במידה והלקוח מתחרט וברצונו להסיר פריט מהעגלה, הוא יכול לבצע זאת בעזרת כפתור remove הנמצא ליד כל פריט אשר נוסף לעגלה.

את כל זה ביצעתי באמצעות פונקציות בקובץ ה CRUD.

3. חיבור לבסיס נתונים ושאילתות SQL

הקובץ ליצירת הטבלאות והכנסת הנתונים אליהם נקרא SqlQueries . יש להריץ את על הטבלאות והשורות בדאטה-בייס ולאחר מכן ניתן להיכנס לאתר ולבחור את הפריטים.

בסיס הנתונים מכיל בתוכו 4 טבלאות :

<u>Items:</u>

טבלה מובנת עם כל הפריטים המוצגים באתר, ממנה מתבצעת רק שאילתת שליפת הפריטים.

Select query- שליפת הפריטים השמורים בטבלה:

```
const sql = require('./db/db');

const getItems = (cb) => {
   const itemsQuery = "select * from items ";
   sql.query(itemsQuery, (err, items) => {
      if (items) {
         cb(items);
      } else {
         cb([]);
      }

   });
}
```

:Cart

טבלה דינמית, מגיעה ריקה מאחר והיא מכילה את הפריטים אשר הלקוח מוסיף לעגלה. השאילתות המתבצעות על טבלה זו הן insert, update, delete, select , אשר בעזרתן הלקוח יוצר רשומה חדשה בעגלה, מעדכן את העגלה על ידי הוספה של פריט אשר כבר קיים בעגלה או מוחק פריט מהעגלה.

Select query- שולף את הפריטים בשמורים בטבלת cart

```
const getCart = (cb) => {
   const itemsQuery = "select * from cart";
   sql.query(itemsQuery, (err, results) => {
      cb(results || []);
   });
}
```

Insert & update queries- יצירת רשומה חדשה בעגלה ועדכון העגלה על ידי הוספה של פריט אשר כבר קיים בעגלה.

```
         const addToCart = (item, cb) => {

   const itemsQuery = "select * from items where id=?";
   sql.query(itemsQuery, item.id, (err, results) => {
      const checkIfExists = "select * from cart where itemId=?";
      const dbItem = results[0];
      sql.query(checkIfExists, dbItem.id, (err, items) => {
         if (items.length) {
            const updated = { ...items[0], quantity: items[0].quantity + 1, price:
items[0].price + item.price }
            update(updated, cb);
         } else {
            add(dbItem, cb);
         };
      });
   });
}

const update = (item, cb) => {
   const updateCartQuery = "update cart set quantity=?, price = ? WHERE itemId= ? "
   sql.query(updateCartQuery, [item.quantity, item.price, item.itemId], (err,
mysqlres) => {
      getCart(cb);
   });
}

const add = (item, cb) => {
   const cartItem = {
```

```
        price: item.price,
        name: item.name,
        category: item.category,
        itemId: item.id,
        image: item.image,
        quantity: 1,
    };

    const addItemQuery = "insert cart set ?";
    sql.query(addItemQuery, cartItem, (err, mysqlres) => {
        getCart(cb);
    });
}
```

Delete query- מחיקה של פריט הנמצא בעגלה.

```
const removeFromCart = (item, cb) => {
    if (item.quantity == 1) {
        const removeQuery = "DELETE from cart where itemId=?";
        sql.query(removeQuery, item.itemId, (err, mysqlres) => {
            getCart(cb);
        });
    } else {
        const itemsQuery = "select * from items where id=?";
        sql.query(itemsQuery, item.id, (err, results) => {
            if (results.length) {
                const updatedItem = {
                    ...item,
                    quantity: item.quantity - 1,
                    price: item.price - results[0].price
                };
                update(updatedItem, cb);
            } else {
                getCart(cb);
            }
        });
    }

}
```

:Orders

טבלה אשר בה נשמרים כל ההזמנות אשר מתבצעות, כוללת בטוחה את פרטי הלקוח אשר הזמין ואת המחיר הכולל של ההזמנה.

:Details

טבלה אשר מקשרת בין הזמנה לפריטים. בתוכה נשמרים הן ה ID של ההזמנה והן של הפריטים המוזמנים, עם הכמות אשר הוזמנה מכל פריט והמחיר הכולל של אותו הפריט. בעלת מפתח זר של טבלת orders וטבלת items.

Insert query- הכנסת רשומות חדשות לטבלאות Items ו details.

```
const insertOrder = (req, res) => {
  if (!req.body) {
    res.status(400).send({
      message: "content cannot be empty"
    });
    return;
  }

  // bring all items from cart
  const getCartItems = "SELECT * from cart ";
  sql.query(getCartItems, (err, items) => {
    if (err) {
      res.render('purchase');
      return;
    }

    const priceSum = items.reduce((sum, item) => {
      sum += item.price;
      return sum;
    }, 0);

    const newOrderEntry = {
      "mail": req.body.email,
      "firstName": req.body.firstName,
      "lastName": req.body.lastName,
      "address": req.body.address,
      "orderDT": new Date(),
      "tA": req.body.takeAway,
      "comments": req.body.comment,
      "totalPrice": priceSum
    }

    // add order to db
    const insertItemQuery = "INSERT INTO orders SET ? ";
    sql.query(insertItemQuery, newOrderEntry, (err, order) => {
      if (err) {
        res.status(400).send({ message: "error on creating order " + err });
        console.log("error on creating order " + err);
        return;
      }
```

```
      console.log("created new order succesfully " + order);

      const details = items.map(item => ([
        item.itemId,
        order.insertId,
        item.quantity,
        item.price
      ]));

      /////////////////////////////////////////////////////////////////////////////////////////
///
      const insertDetailsQuery = "INSERT INTO `details` (itemId , orderID , quantity , price )
VALUES ?";
      sql.query(insertDetailsQuery, [details], (err, mysqlres) => {
        if (err) {
          console.log("error on getting details " + err);
          res.status(400).send({ message: "error on getting details " + err });
          return;
        }
        console.log("order details saved " + mysqlres);
        sql.query('TRUNCATE TABLE cart', (err, result) => showDetails(newOrderEntry, items
,res));

      });
    });


  });
}
```

Select query- הצגה סיכום של ההזמנה ללקוח בסיום.

```
const showDetails = (orderEntry, cartItems, res) => {
  res.render('purchase-details', {
  orderEntry: {
    ...orderEntry,
    orderDT: new Date(orderEntry.orderDT).toLocaleString(),
    tA: orderEntry.tA == 0 ? 'No' : 'Yes'
  },
  cartItems,
  shoppingCart: []});
}
```

4. מימוש טפסים

טופס ביצוע הזמנה על יד הלקוח. הלקוח ממלא את פרטיו, אינו יכול לבצע הזמנה עד להשלמת מילוי כל הפרטים הנדרשים, כולל תקינות האימייל שלו. במידה והלקוח מעוניין במשלוח יש באפשרותו ללחוץ על כפתור של מציאת המיקום המחזיר את מיקומו בעזרת קווי רוחב ואורך (בחלק ב של הפרויקט הצלחתי להחזיר את הכתובת של המשתמש אך כעת על מנת להחזיר את הכתובת היה עלי למסור פרטי אשראי לגוגל אז נשארתי עם קווי האורך ורוחב). לאחר מכן הלקוח ישלח לעמוד סיכום פרטי ההזמנה.

```
form#form1(action='purchase-details' , method="post")
    label(for='firstName') First Name :
    input#firstName(form='form1' required='' type='text' name='firstName' value='' min='2')
    br
    label(for='lastName') Last Name :
    input#lastName(form='form1' required='' type='text' name='lastName' value='' min='2')
    br
    label(for='email') Email :
    input#email(type='email' required='' name='email' value='')
    br
    input(type='radio' id='delivery' name='takeAway' onchange='showPositionTracker(true)'
value=0)
    label(for='delivery') Delivery
    input(type='radio' id='takeAway' name='takeAway' onchange='showPositionTracker(false)'
value=1 checked)
    label(for='takeAway') Take Away
    button#location-btn(hidden='' onclick=' return getLocation()') Find you&apos;r location
    p#p
    label(for='Address') Address :
    input#address(type='text' required name="address")
    p *Payment is made in cash
    label(for='comment') If you have any comment, please write it here
    br
    textarea#comment(name='comment' cols='30' rows='5')
    br
    input(type='submit' value='Approve')
```

5. מימוש פונקציונאליות עיבוד מידע

בחלק זה מימשתי פונקציה שלוקחת מידע מהלקוח באמצעות טופס, מעבדת את המידע ומחזירה תוצאה.

לאחר בחירת הפריטים אותם הלקוח רוצה להזמין, הוא ממלא טופס המכיל את פרטיו על מנת להשלים את תהליך ההזמנה. המערכת מושכת את הפרטים שהוזנו בעת שליחת הטופס ופותחת דף חדש למשתמש עם סיכום כל פרטי ההזמנה, הכולל את פרטיו, ופירוט המאפים אשר בחר להזמין עם הכמות והמחיר הכולל של ההזמנה.

6. מימוש view engine

בחרתי להשתמש בקבצי pug.
יצרתי קובץ layout כללי אשר מכיל את מבנה האתר, תפריט משמאל, התוכן באמצע והעגלה מימין.
את שאר עמודי האתר אשר היו קיימים כבר כקובץ html המרתי לקבצי pug כאשר הם מכילים רק את התוכן, החלק האמצעי במבנה בכל עמוד.
העמודים הקיימים הם: index, purchase, purchase_detail, about.