פרויקט קורס WEB – חלק ג

קבוצה 2

מגישים:

עומרי חיאל	308451699
אורין נוריאל	205768849
מירה חורי	206952863
כליל להב	205888753
סער חן	308437847

תוכן

הקדמה

בחלק זה של הפרויקט שמנו דגש על server-side [לעומת חלק ב, שבו התעסקנו בנראות האתר וב client-side]. בחלק זה בנינו את בסיס הנתונים של האתר שלנו וקישרנו בין הטפסים שממולאים ע"י לקוחות האתר לבין מאגר הנתונים שלנו.

בחלק זה עבדנו בעיקר עם שאילתות SQL על מנת להפוך את מילוי מאגר הנתונים וכן גם את חווית המשתמש לנוחה יותר, זאת ע"י מתן אפשרויות סינון וחיפוש מידע. בעזרת ממשק ה-PUG הצלחנו לייבא ולייצא את הנתונים בצורה נוחה יותר בעבור הclient-side.

באתר ניתן למצוא עמודי מכירה עם כל פרטי הכרטיס הנמכר, עמודי רכישה לפי קטגוריה ואפשרויות חיפוש לפי שם/תאריך המופע המבוקש. וכן, בעת ריחוף מעל כל כרטיס יינתן חיווי ובעת לחיצה יופיע עמוד המכיל את פרטי הכרטיס המלאים.

בנוסף לכך, בשל העובדה כי אתרינו מתעסק בעיקר במכירה ובקניה של כרטיסים, בעבור כל משתמש רשום (בנוסף לחיווי שהוא אכן מחובר לאתר) נוסף דף פרופיל אשר בו תוצג כל היסטורית פעילותיו באתר; כרטיסים שמכר, כרטיסים שמוצעים למכירה וכרטיסים שהמשתמש רכש.

קובץ server.js- מקבל בתוכו את המודלים הרלוונטיים שהותקנו (מפורט בארגון תיקיות הפרויקט) כמו כן במודולים שיצרנו:DB.js & crud operations, הSERVER מכיל את הroutes לכל העמודים והפונקציות והינו מאזין לport 3000.

ארגון תיקיות הפרויקט

על מנת לעבוד בצורה מסודרת יותר החלטנו לארגן מחדש את תיקיות הפרויקט ולעבוד בצורה המקובלת בשוק. העבודה נעשתה ע"י חלוקה לתיקיות לפי הסדר הבא:

- App-route directory-(include server.js, package.json, package-lock.json)
    - bin-
    - DB-
        - DB(connection to DB)
        - DB config (connection details to DB)
        - DB script (this file need to use for our web. need to run on command)
    - Public
        - img(מכיל את כל התמונות שהשתמשנו בהם בבניית האתר)
        - stylesheets(including css files)
        - javascripts(including javascript files)
    - node_modules(also include models we used :node js, express, mysql2 bodyparser)
    - routes (include crud_opration.js (all routes functions))
    - view (include all PUG pages\ web pages)

הנחות

<u>הנחות נוספות לחלק ג:</u>

▪ בעת הוספת כרטיס, נניח שמצרפים רק קבצי pdf או img של הכרטיס המוצע למכירה.

▪ החלטנו בERD שאין צורך בישויות: event, place ואת הפרטים הרלוונטיים מישויות אלו העברנו לטבלת tickets. וכן ויתרנו על שדות מסוימים והכנסו רק את אלה הרלוונטיים.

▪בעדכון פרטים יצרנו דוגמא לעדכון, ניתן לעדכן את הסיסמא בלבד.

חיבור לבסיס נתונים ושאילתות SQL

בחלק זה נציג את הטבלאות הנמצאות בבסיס הנתונים:

1. טבלת משתמשים-(USERS) הטבלה מכילה את העמודות הבאות: שם משתמש, סיסמא, מספר טלפון ותאריך הצטרפות לאתר.
2. טבלת קטגוריות- (CATEGORIES)טבלה זו משמשת כ lookup-table
3. טבלת כרטיסים- (TICKETS)בטבלה זו מוצגים כל פרטי המכירה הרלוונטיים לכרטיס.
4. טבלת עסקאות- (DEALS)בטבלה זו מופיעים מספר הכרטיס הנמכר, פרטי המוכר ופרטי הקונה.
5. טבלת יצירת קשר-(COTACTS) טבלה זו כוללת את הטפסים שנשלחו עבור התקשרות ובינהן שם המבקש, האימייל שלו ומה תוכן ההודעה.
6. טבלת התחברויות (CONNECTIONS)-טבלה ששומרת את ההתחברויות האחרונות של המשתמשים לפי שם ושעת כניסה.

<u>לצפייה בטבלאות</u>

בנוסף לכך, נעזרנו בטבלאות view :

- טבלה בשם aviable-ticket המציגה את הכרטיסים הזמינים לרכישה בלבד.

- וטבלת user-ticket שהיא חיבור ע"י Join של טבלאות כך שתיווצר חלוקה אילו פריטים רכש, אילו פריטים קנה ואילו פריטים הוא מציע למכירה על מנת לסווג ולהציג זאת בעמוד my_profile בצורה מאורגנת.

שאילתות SQL:

בחירה-

בעבור עמוד tickets השתמשנו בשאילתה אשר מציגה את כל הכרטיסים הזמינים לנו כעת (הצגת כל השורות של טבלת avileble_tickets).

הוספה-

עמוד הוספת כרטיס (add_tickets) מתבצעת הוספת כרטיס לטבלת הכרטיסים (tickets).
עמוד הרשמה (registration)- בעמוד זה ע"י קבלת מידע מהמשתמש נוסיף ע"י שאילתה את פרטי המשתמש החדש לטבלת users

עמוד כניסה(sign in)-לטבלת connections תתוסף רשומה חדשה עם שם המשתמש שנכנס והזמן בו נכנס.

עמוד רכישת כרטיס(ticket_page) – מעמוד זה תתבצע הוספת רשומה חדשה לטבלת DEALS.

עדכון-

בעבור כרטיסים שנקנו יתקיים עדכון לטבלת tickets לכך שהכרטיס אינו זמין בשדה available.

מחיקה-

ביצעו אופציה למחיקת המשתמש מתוך הDB.

*כמו כן קיימות עוד שאילתות רבות אשר שימשו אותנו לצרכים שונים למימוש טפסים באתר שלנו (יפורט בהמשך).

מימוש טפסים

מרבית מהטפסים אותם נציין בהמשך נוצרו באתר בעת ביצוע חלק ב של הפרוייקט. כעת נפרט על מימושם, ברובם עי" לחיצת על כפתור המימוש מופעלת פונקציה אשר שולחת route לserver.js וכך גורמת לקישור בין הפרטים שהוזנו לבין הDB, הפונקציות מוסיפות רשומות חדשות לטבלאות הנדרשות, בנוסף לצורך מימוש הטפסים והכנסת לDB השתמשנו במטודת POST של EXPRESS ובשביל לקבל מידע מהDB השתמשנו במטודת GET של EXPRESS.

טופס העלאת כרטיס-

אנו יוצרים אובייקט אשר השדות בו זהים לשדות בטבלת tickets, המימוש מתבצע הן מבחינת הוספת הכרטיס לDB באמצעות POST והן מבחינת הוספתו באופן ויזואלי לאתר באמצעות GET.

טופס הרשמה-

בחלק זה הטופס מומש כך שבעת הזנת פרטי המשתמש ושיגור הבקשה להרשמה לאתר תתווסף רשומה חדשה בDB בטבלת USERS המימוש מתבצע באמצעות POST.

טופס התחברות-

בטופס זה תתבצע הוספה של רשומה באמצעות POST לטבלת CONNECTIONS

טופס יצירת קשר-

בטופס זה תתבצע הוספה של רשומה באמצעות POST לטבלת CONTACT.

טופס עדכון פרטי משתמש (סיסמא)-

בטופס זה מתבצע עדכון בעבור סיסמת המשתמש ע"י POST לטבלת USERS.

מימוש פונקציה

בעבור הצגת הכרטיסים במקומות הרלוונטים כמו בעמוד הבית, עמוד הפרופיל שלי ובעבור עמוד הכרטיסים נוצרו פונקציות אשר ע"י שאילתות ותנאים מסויימים מכניסים את הכרטיסים הרלוונטים לתוך מערכים ומתוך המערכים האלו נוספים הכרטיסים הרלוונטים למקומות הויזואלים הרלוונטים.

:home_page

הגדרנו כי כרטיסים יוצגו לפי הקטגוריות הבאות כרטיסים להיום, כרטיסים שהועלו לאחרונה, וכרטיסים בהנחה.

בשביל שכרטיס יכנס למערך אנחנו נבחר את כל הכרטיסים מטבלת avilableTickets ובעצם ניצור פילטרים כך שעבור המערכים הבאים

todayEventsList- נוצר תנאי שרק כאשר התאריך של האירוע (event_date) זהה לתאריך של היום הכרטיס יכנס למערך.

lastUploadsList-בשאילת הוגדר כי הכרטיסים יופיעו לפי סידור של זמן העלאה מגדול לקטן ובעצם למערך יתווספו רק 10 הראשונים.

discountTicketsList- נוצר תנאי אשר רק כאשר sale_price קטן מ purches_price הוא יכנס למערך הנוכחי

:My_profile

הגדרנו כי עבור כל משתמש יוצגו כרטיסים ע"י הקטגוריות הבאות כרטיסים שמכר, כרטיסים שקנה וכלל הכרטיסים שהעלה.

לצורך כך השתמשנו בשאילתה אשר לוקחת את הנתונים מטבלת ticketUsers ונוצרו פילטרים למערכים כך ש:

forSale- נוצר תנאי שמשווה בין הID של המשתמש המחובר(active user) כעת לowner ורק כאשר הם שווים מכניס אותו למערך של הכרטיסים המיועדים וזמינים למכירה.

Sells- התנאי שנוצר הוא שהID של המשתמש המחובר(active user) כעת צריך להיות שווה לseller. ולאחר מכן מכניס למערך.

Purcheses- התנאי הוא השוואה בין הID של המשתמש המחובר(active user) כעת לbuyer. ולאחר מכן אובייקט זה יכנס למערך הכרטיסים.

ה-active user- השם של המשתמשל של המשתמש המחובר כעת, זהו משתנה גלובלי כאשר הנמשתמש מחובר כל הפעולות שיבצע באתר יקרו בשמו, כאשר המשתמש מתנק המשתנה הגלובלי הוא משתנה לguest.

tickets:

השאילה בעבור עמוד זה משתמשת בכל הכרטיסים מטבלת availableTickets ומכניסה אותם למערך המוצג לפי סדר האלפבית בעמוד זה.

לקוד של הפונקציות

תוכן דינמי

על מנת שעמודי האתר שלנו יתעדכנו בהתאם לשינויים שנעשים בטבלאות שבבסיס הנתונים שלנו המרנו את כל עמודי הHTML לעמודי PUG, כעת העמודים יכולים להתעדכן באופן אוטומטי מהDB ע"י משתנים שהגדרנו ופונקציות שבנינו ב-crud_operations .

בצורה זאת חסכנו בשורות קוד רבות הצלחנו לסדר את האתר בצורה שמתבססת על הDB

ובאופן דינמי.

נספחים

נספח חיבור לבסיס נתונים ושאילתות SQL:

כל הטבלאות שיש בבסיס הנתונים

Tables_in_web_project_g2
availabletickets
categories
connections
contacts
deals
tickets
users
usertickets

עבור טבלת avilabletickets

Field	Type	Null	Key	Default	Extra
ticket_ID	int	NO		0	
event_name	varchar(255)	NO		NULL	
category	varchar(255)	YES		other	
event_date	date	NO		NULL	
start_time	time	NO		NULL	
location_name	varchar(255)	NO		NULL	
address	varchar(255)	NO		NULL	
row_seat	int	YES		NULL	
chair	int	YES		NULL	
purchase_price	double	NO		NULL	
sale_price	double	NO		NULL	
qunatity	int	YES		1	
phone	varchar(10)	YES		NULL	
sold	tinyint(1)	YES		0	
owner	varchar(255)	NO		NULL	
upload_time	datetime	NO		NULL	

עבור טבלת categories

Field	Type	Null	Key	Default	Extra
category	varchar(255)	NO	PRI	NULL	

עבור טבלת connections

Field	Type	Null	Key	Default	Extra
username	varchar(255)	NO	PRI	NULL	
connection_time	datetime	NO	PRI	NULL	

עבור טבלת contacts

Field	Type	Null	Key	Default	Extra
contact_ID	int	NO	PRI	NULL	auto_increment
full_name	varchar(255)	NO		NULL	
phone_number	varchar(10)	NO		NULL	
email	varchar(255)	NO		NULL	
content	varchar(500)	YES		NULL	
contact_time	datetime	YES		NULL	

עבור טבלת deals

Field	Type	Null	Key	Default	Extra
deal_ID	int	NO	PRI	NULL	auto_increment
ticket_ID	int	NO		NULL	
ticket_price	double	YES		NULL	
seller	varchar(255)	YES	MUL	NULL	
buyer	varchar(255)	YES	MUL	NULL	
deal_datetime	datetime	NO		NULL	

עבור טבלת tickets

Field	Type	Null	Key	Default	Extra
ticket_ID	int	NO	PRI	NULL	auto_increment
event_name	varchar(255)	NO		NULL	
category	varchar(255)	YES	MUL	other	
event_date	date	NO		NULL	
start_time	time	NO		NULL	
location_name	varchar(255)	NO		NULL	
address	varchar(255)	NO		NULL	
row_seat	int	YES		NULL	
chair	int	YES		NULL	
purchase_price	double	NO		NULL	
sale_price	double	NO		NULL	
qunatity	int	YES		1	
phone	varchar(10)	YES		NULL	
sold	tinyint(1)	YES		0	
owner	varchar(255)	NO	MUL	NULL	
upload_time	datetime	NO		NULL	

עבור טבלת users

Field	Type	Null	Key	Default	Extra
username	varchar(255)	NO	PRI	NULL	
email	varchar(255)	NO		NULL	
password	varchar(255)	NO		NULL	
phone_number	varchar(10)	YES		NULL	
join_date	date	NO		NULL	

עבור טבלת usertickets

Field	Type	Null	Key	Default	Extra
ticket_ID	int	NO		0	
event_name	varchar(255)	NO		NULL	
category	varchar(255)	YES		other	
event_date	date	NO		NULL	
start_time	time	NO		NULL	
location	varchar(255)	NO		NULL	
address	varchar(255)	NO		NULL	
price	double	NO		NULL	
owner	varchar(255)	NO		NULL	
seller	varchar(255)	YES		NULL	
buyer	varchar(255)	YES		NULL	

לחזרה לדוח

נספח מימוש פונקציה:

Tickets

```
const getAvailableTickets = function(req, res) {
    var ticketsList = [];
    sql.query('SELECT * FROM availableTickets', function(err, rows) {
        if (err) {
            res.status(500).json({"status_code": 500,"status_message": "internal server error"});
        } else {
            // Loop check on each row
            for (var i = 0; i < rows.length; i++) {

                // Create an object to save current row's data
                var ticket = {
                    'ticket_ID':rows[i].ticket_ID,
                    'event_name':rows[i].event_name,
                    'location_name':rows[i].location_name,
                    'address':rows[i].address,
                    'purchase_price':rows[i].purchase_price,
                    'event_date':rows[i].event_date,
                    'start_time':rows[i].start_time,
                    'category':rows[i].category
                }
                // Add object into array
                ticketsList.push(ticket);
                console.log('titcket added to array');
            }
        console.log(ticketsList)
        // Render tickets.pug page using array
        res.render('tickets', {"ticketsList": ticketsList});
        }
    });
}
```

My profile

```
const getMyProfile = function(req, res) {
    if (active_user == 'guest'){
        res.render('sign in');
    }
    var forSale = [];   //array of all the tickets that the active user uploaded and offers for sale
    var sells = [];     //array of all the tickets that the active user already sold
    var purchases = []; //arry of all the tickets that the active user bought
    sql.query('SELECT * FROM userTickets', [active_user], function(err, rows) {
        if (err) {
            res.status(500).json({"status_code": 500,"status_message": "internal server error"});
        } else {
            // Loop check on each row
            for (var i = 0; i < rows.length; i++) {
                if (active_user == rows[i].owner){
                    // Create an object to save current row's data
                    var saleTicket = {
                        'ticket_ID':rows[i].ticket_ID,
                        'event_name':rows[i].event_name,
                        'location_name':rows[i].location,
                        'address':rows[i].address,
                        'purchase_price':rows[i].price,
                        'event_date':rows[i].event_date,
                        'start_time':rows[i].start_time,
                        'category':rows[i].category
                    }
                    // Add object into array
                    forSale.push(saleTicket);
                    console.log('titcket added to array');
                }
            }
```

```
            console.log(forSale);

            for (var i = 0; i < rows.length; i++) {
                if (active_user == rows[i].seller) {
                    // Create an object to save current row's data
                    var sellTicket = {
                        'ticket_ID':rows[i].ticket_ID,
                        'event_name':rows[i].event_name,
                        'location_name':rows[i].location,
                        'address':rows[i].address,
                        'purchase_price':rows[i].price,
                        'event_date':rows[i].event_date,
                        'start_time':rows[i].start_time,
                        'category':rows[i].category
                    }
                    // Add object into array
                    sells.push(sellTicket);
                    console.log('titcket added to array');
                }
            }
            console.log(sells);

            for (var i = 0; i < rows.length; i++) {
                if (active_user == rows[i].buyer) {
                    // Create an object to save current row's data
                    var buyTicket = {
                        'ticket_ID':rows[i].ticket_ID,
                        'event_name':rows[i].event_name,
                        'location_name':rows[i].location,
                        'address':rows[i].address,
```

```
                        'address':rows[i].address,
                        'purchase_price':rows[i].price,
                        'event_date':rows[i].event_date,
                        'start_time':rows[i].start_time,
                        'category':rows[i].category
                    }
                    // Add object into array
                    purchases.push(buyTicket);
                    console.log('titcket added to array');
                }
            }
            console.log(purchases);
            res.render('my_profile', {"forSale": forSale, "sells": sells, "purchases": purchases});
        }
    });
}
```

Home page

```
//get data of the tickets in the home page
const getHomePage = function(req, res) {
    var todayEventsList = [];       //array of all the ticket that occurs today
    var lastUploadsList = [];       //array of the 10 last uploaded tickets to the website
    var discountTicketsList = [];   //array of the tickets that sells at lower price then it boughts
    sql.query('SELECT * FROM availableTickets order by upload_time desc', function(err, rows) {
        if (err) {
            res.status(500).json({"status_code": 500,"status_message": "internal server error"});
        } else {
            // Loop check on each row
            //today
            for (var i = 0; i < rows.length; i++) {
                if (today() == rows[i].event_date){
                    // Create an object to save current row's data
                    var todayEvent = {
                        'ticket_ID':rows[i].ticket_ID,
                        'event_name':rows[i].event_name,
                        'location_name':rows[i].location,
                        'address':rows[i].address,
                        'purchase_price':rows[i].price,
                        'event_date':rows[i].event_date,
                        'start_time':rows[i].start_time,
                        'category':rows[i].category
                    }
                    // Add object into array
                    todayEventsList.push(todayEvent);
                    console.log('titcket added to array');
                }
            }
            console.log(todayEventsList);
```

```
            //upload
            for (var i = 0; i < 10; i++) {
                // Create an object to save current row's data
                var uploadTicket = {
                    'ticket_ID':rows[i].ticket_ID,
                    'event_name':rows[i].event_name,
                    'location_name':rows[i].location,
                    'address':rows[i].address,
                    'purchase_price':rows[i].price,
                    'event_date':rows[i].event_date,
                    'start_time':rows[i].start_time,
                    'category':rows[i].category
                }
                // Add object into array
                lastUploadsList.push(uploadTicket);
                console.log('titcket added to array');
                if (i+1 == rows.length){
                    break;
                }
            }
            console.log(lastUploadsList);
            //discount
            for (var i = 0; i < rows.length; i++) {
                if (rows[i].sale_price < rows[i].purchase_price) {
                    // Create an object to save current row's data
                    var discountTicket = {
                        'ticket_ID':rows[i].ticket_ID,
                        'event_name':rows[i].event_name,
                        'location_name':rows[i].location,
                        'address':rows[i].address,
```

```
                        'purchase_price':rows[i].price,
                        'event_date':rows[i].event_date,
                        'start_time':rows[i].start_time,
                        'category':rows[i].category
                    }
                    // Add object into array
                    discountTicketsList.push(discountTicket);
                    console.log('titcket added to array');
                }
            }
            console.log(discountTicketsList);
            res.render('home_page', {"todayEventsList": todayEventsList,"lastUploadsList": lastUploadsList,
                "discountTicketsList":discountTicketsList, 'active_user': active_user});
        }
    });
}
```