הפקולטה למדעי ההנדסה

המחלקה להנדסת תעשייה וניהול

שם הקורס : בניית מערכות ממוחשבות מבוססות web

פרויקט הקורס

חלק 3 - צד שרת

קבוצת הגשה: קבוצה מספר 1

שמות ות.ז המגישים:

גל ביטון 203106166

רון גבו 313194557

שקד פירנקו 314762444

חן הרשקו 313444044

20.02.2022

תוכן עניינים

1. הנחות

- פרטי הלקוחות שמורים באופן מאובטח (הטרנזקציות נשמרות כרגע בבסיס הנתונים, באתר שמחייב באמת זה היה מתבצע באמצעות חברת סליקה ולא נשמר שם, השמירה היא רק בשביל ההדמייה).
- לא ניתן לקבוע תור למספרה באותו יום, אלא רק באמצעות התקשרות למספר הרשום. זאת כיוון שספי הספר צריך לדעת יום מראש איך לתכנן את היום ולסנכרן בין מי שנרשם דרך האתר ומי שלא
- כאשר לקוח ממלא טופס "צור קשר", אחד מעובדי המספרה חוזר אליו עד כשני ימי עסקים.

2. חבילות

במהלך הפרוייקט השתמשנו במספר חבילות, בנוסף לחבילות שנלמדו בקורס. יצרנו קובץ requirements.txt שיכיל את החבילות הללו עבור החבילות בפייתון:

- Flask - עבדנו עם הגרסה 2.0.2
- Mysql-connector-python - עבדנו עם הגרסה 8.0.27
- Python-dotenv - עבדנו עם הגרסה 0.19.2

עבור JAVASCRIPT, השתמשנו גם בייבוא של:

- JQuery
- Bootstraps

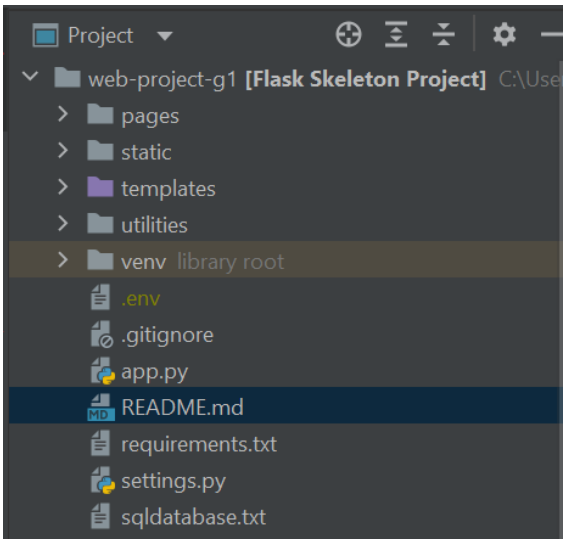
3. מבנה הפרויקט

3.1. ארגון מחדש

בחלקו השני של הפרויקט עבדנו על הצד לקוח של האתר. ארגנו מחדש את השלד של הפרויקט למבנה של השלד של פרויקט FLASK שניתן לנו במודל.

3.2. מבנה

מבנה התיקיות החיצוני:



Pages - נפרט על המבנה הזה יותר בהמשך, כמו השלד שניתן זה מכיל את כל הBLUEPRINTS עם הדפי ROUTES/CSS/HTML שלהם.

Static- נפרט על המבנה הזה בהמשך. מכיל קבצים סטטים של IMAGES/WEBFONTS/JS/CSS שמשתמשים בשלד של דף HTML שלנו.

Templates - נפרט על המבנה הזה בהמשך. מכיל קבצי HTML גלובליים לכל הפרוייקט

Utilities - נפרט על המבנה הזה בהמשך. הוא מכיל את כל הקוד שמטפל עם הDATABASE

.env - מכיל מידע של הסביבה להרמת השרת והתחברות לדטא בייס.

.gitignore - מכיל קבצים מהם נרצה לא להעלות לגיטהאב, בין היתר env. וכו'.

App.py – השרת של הפרוייקט, הוא רושם את כל הBLUEPRINTS מכל הדפים, מרים את השרת, ועוד.

README.md – קובץ רידמי שהגיע עם שלד הפרוייקט.

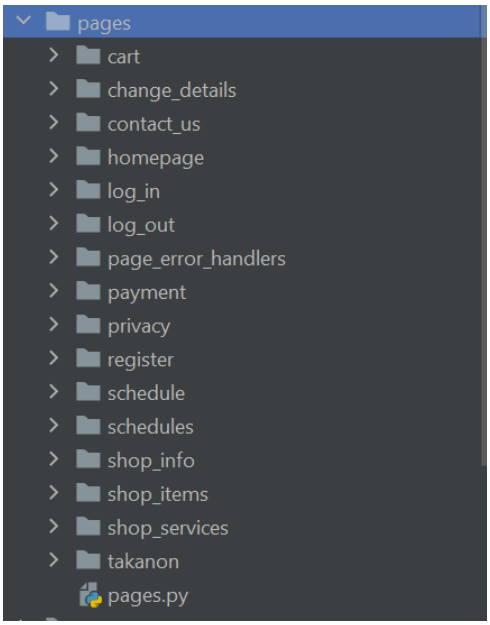
Requirements.txt – קובץ שמכיל את החבילות והגרסאות שהשתמשנו בהן בפייתון שהן לא חלק מהספרייה הסטנדרטית של השפה

Settings.py – קובץ הגדרות שהגיע עם שלד הפרוייקט

Sqldatabase.txt – מכיל את כל הפקודות שביצענו על מנת לייצר את הדטה בייס לאתר.

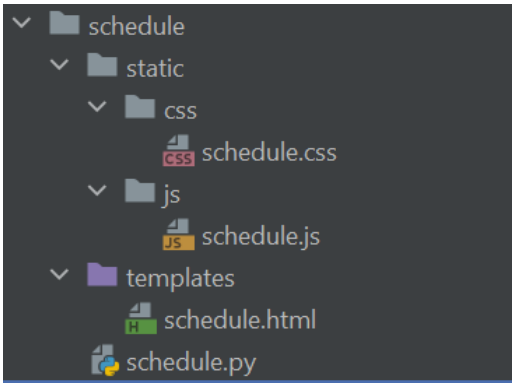
Pages

מכיל תיקיות, כל תיקיה במבנה של שלד של פרוייקט פלאסק.



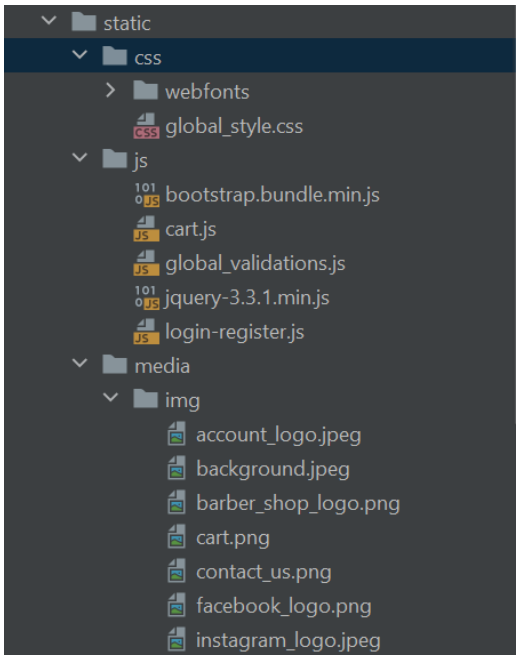
כל תיקייה מייצגת לרוב עמוד (אם לפעמים רק ROUTES), כאשר כל תיקייה בpages בעלת קובץ פייתון שמייצר ROUTES, ותיקיות JS/HTML/CSS מתאימות

דוגמא למבנה של תיקייה פנימית :



Static

מכיל קבצי IMAGES/JS/CSS שבשימוש ביותר מדף אחד.

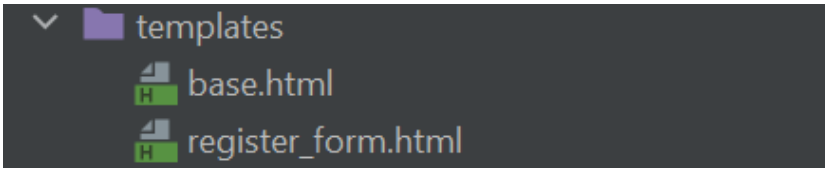


תיקיית CSS – CSS שהוא בשימוש ע"י שלד של דף באתר (ע"י base.html – נגיע אליו בהמשך)

תיקיית JS – מכילה קוד JS שבשימוש ביותר מדף אחד

- Cart.js - לוגיקה של העגלת קניות. מחשבת בכל דף שנמצאים את סה"כ הסכום כרגע, שהעגלה לא ריקה לפני שעושים צ'ק אאוט ועוד.
- Global_validations.js – קובץ שמכיל המון ולידציות שבשימוש נרחב, לדוגמא ולידציות לתאים של טפסים (כמו ולידציה על אימייל, טלפון וכו')
- Login-register.js – קוד JS שיודע להוריד ולהעלות את הPOPUP של ההרשמה וההתחברות בעת לחיצה על הרשמה/התחברות או לחיצה על האיקס בחלון שנסגר
- שני הקבצים האחרים הם ייבוא ע"מ להשתמש בפונקציונליות שלהן (JQUERY, BOOTSTRAP)

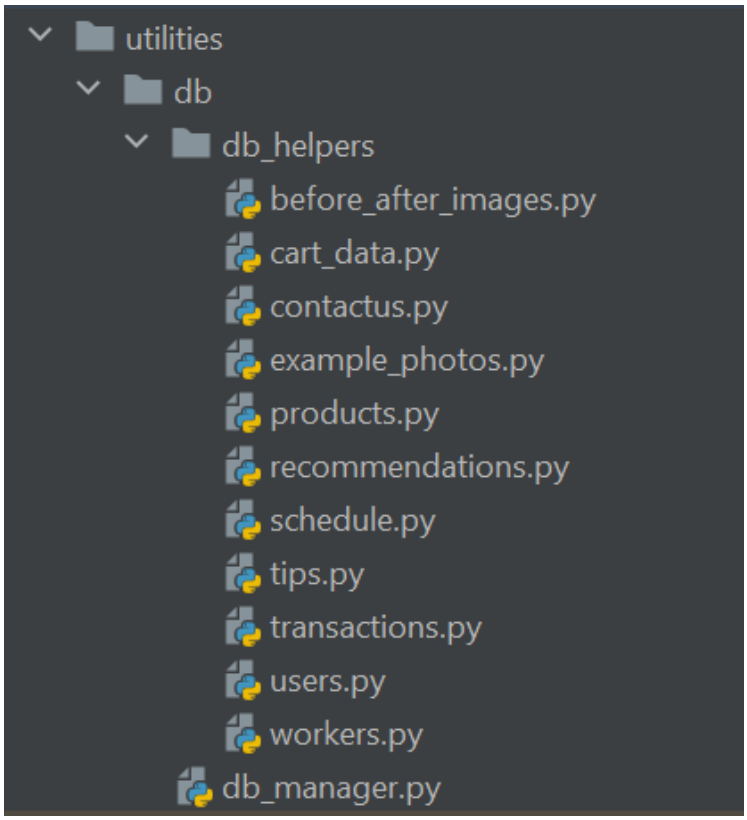
Templates



- Base.html, שלד של דף באתר.
- Register_form.html – טופס של הרשמה, בשימוש ביותר מדף אחד ולכן שמנו אותו כאן.

Utilities

מכיל את כל הקוד להתממשק עם הטבלאות בבסיס הנתונים.



- Db_manager , קטע קוד שקיבלנו עם השלד שמבצע את הקריאות לדטה בייס. הוספנו לו פונקציות גנריות
  - Build_fetch_query
  - Build_insert_query
  - Build_update_query

שיוצרות שאילתא ע״מ להימנע מהרבה קוד כפול (עבור DELETE רק 2 שאילתות נוצרו ולכן לא יצרנו פונקציה גנרית עבור שאילתא מסוג DELETE).

- לכל טבלה בDATABASE יצרנו קובץ פייתון בdb_helpers שמתממשקת עם הטבלה ומחזירה את המידע מהטבלה ע״י שימוש בdbManager.

4. חיבור לבסיס הנתונים ושאילתות SQL

4.1. בסיס נתונים

יצרנו תשתית לצד השרת באמצעות תוכנת MySQL. התכנה MySQL הינו מסד נתונים יחסי, רב נימי ורב משתמשים מבוסס שפת SQL. בסיס הנתונים שלנו נקרא web-project-g1.

4.2. הטבלאות

Contact_us

טבלה ששומרת את הבקשות של הלקוחות ליצירת קשר,

- Phone – מספר הטלפון של הפונה במידה והשאיר
- Email - אימייל של הפונה במידה והשאיר
- Contact_name – שם הפונה במידה והשאיר
- Contact_description – תיאור סיבת הפנייה, במידה והשאיר

ישנה ולידציה לפני הכנסה לטבלה כי לפחות הפלאפון או האימייל נתנו, זאת על מנת שנוכל לחזור ללקוח

Products

טבלה ששומרת את כל המוצרים, כולל כמה נקנה מכל מוצר.

- Product_id – זהו PK אשר מג'ונרט באופן אוטומטי ועולה בצורה אינקרמנטלית.
- Price – מחיר המוצר
- Image_url – לינק של התמונה של המוצר
- Num_bought – מספר הפעמים שהמוצר הנ"ל נקנה

Tips

טבלה ששומרת את כל הטיפים שיוצגו באתר. הטיפים הוכנסו לטבלה על מנת שנוכל לייצר סקשן דינאמי של HTML עם טיפים שנוכל לשנות בלי צורך לעשות ריסטרט לשרת.

- Tip – מחרוזת של הטיפ.

Recommendations

המלצות של הלקוחות על ספי הספר. נכנס על מנת שנוכל לשלוט בכמות ההמלצות בצורה דינאמית

- Recommendation – מחרוזת של ההמלצה על ספי הספר
- Created_date – תאריך בו נוצרה השורה בבסיס נתונים, נוצר באופן דיפולטי ברגע ההכנסה.

Before_after_photos

תמונות של לפני ואחרי הטיפולים של ספי הספר, נכנס לבסיס נתונים על מנת שנוכל לשלוט בכמות התמונות בצורה דינאמית

- Image_url – לינק לתמונה של הלפני ואחרי

Example_photos

תמונות לדוגמא של התספורות שביצע ספי הספר. נכנס לבסיס נתונים על מנת שנוכל לשלוט בתמונות הללו בצורה דינאמית

- Image_url – לינק לתמונה של דוגמא של טיפול של ספי הספר

Workers

טבלה המכילה מידע על העובדים במספרת "ספי הספר", משמשת בין היתר לדף shop_info שמספר על העובדים.

- Worker_name – שם העובד
- Image_url – לינק לתמונה של העובד
- Worker_description – תיאור של העובד

Users

טבלה השומרת מידע על המשתמשים של האתר.

כל משתמש שנרשם מתווסף לטבלה הנ"ל עם פרטי ההרשמה

- Email – אימייל הנרשם, PK, לא ניתן כפילויות של אימיילים בין שורות שונות
- Password – סיסמא לחשבון
- First_name – שם פרטי של המשתמש
- Last_name – שם המשפחה של המשתמש

Cart_data

טבלה השומרת את המידע על עגלות הקניות באתר. כל שורה מכילה את האימייל של המשתמש אליו הפריט שייך, הID של המוצר אותו הוסיף, ותאריך שבו שילם על המוצר הנ"ל (אם עדיין לא שילם הערך יהיה NULL)

- User_email – האימייל של המשתמש אשר המוצר בעגלה שייך לו, TR FK לemail מטבלת users.
- Product_id – המזהה הייחודי של המוצר, FK לproduct_id מטבלת products
- Closed_session_date – התאריך בו הלקוח שילם על המוצר הנ"ל

חשוב לציין שבמידה והמשתמש מחליט לשנות את האימייל איתו נרשם, או שהID של מוצר כלשהו משתנה, הטבלה הזאת מעודכנת אוטומטית כיוון שהוספנו ביצירת עמודות את ON UPDATE CASCADE.

Transactions

מכיל מידע על כל ההעסקאות שבוצעו באתר.

הטבלה באה לדמות את המצב שבו יש חברת סליקה אשר מבצעת את העסקה ושומרת את פרטי התשלום במקום כלשהו, ברור כי שמירה אמיתית של פרטים אלו בבסיס נתונים איננה בטוחה, הטבלה רק לשם ההדמיה

- Cc_number – מספר כרטיס אשראי
- User_id – מספר ת"ז של המשתמש
- Cvv – 3 ספרות בגב הכרטיס
- Total – כמה סה"כ שולם
- expDate – תאריך פג תוקף כרטיס אשראי

Schedule

טבלה המכילה את כל התורים שנקבעו למרפאה ע"י האתר.

- Schedule_time – תאריך קבעית התור.
- Phone – מס' הטלפון של קובע התור.
- Schedule_type – סוג הטיפול – החלקה, תספורת וכו'.
- Email - אימייל של קובע התור.

4.3. שאילתות

לנוחיות, הגדרנו את הפונקציות הגנריות הבאות לביצוע השאילתות בdbManager

```
def build_fetch_query(self, table_name, table_columns=None, conditions=None, limit=None, order_column=None,
                      order_type='ASC'):
    select_columns = f'{",".join(table_columns)}' if table_columns else '*'
    where_clause = f'WHERE {" AND ".join(conditions)}' if conditions else ''
    order_clause = f'ORDER BY {order_column} {order_type}' if order_column else ''
    limit_clause = f'LIMIT {limit}' if limit else ''
    return self.fetch(f'SELECT {select_columns} from {table_name} {where_clause} {order_clause} {limit_clause};

def build_insert_query(self, table_name, table_columns, values):
    values_clause_list = [f'({",".join(value)})' for value in values]
    values_clause = ','.join(values_clause_list)
    return self.commit(f'INSERT INTO {table_name} ({",".join(table_columns)}) VALUES {values_clause};')

def build_update_query(self, table_name, updates: dict, conditions=None):
    update_clauses = ', '.join(f'{col} = {val}' for col, val in updates.items())
    set_clause = f'SET {update_clauses}'
    where_clause = f'WHERE {" AND ".join(conditions)}' if conditions else ''
    return self.commit(f'UPDATE {table_name} {set_clause} {where_clause};')
```

שאילתות מסוג FETCH

```
@staticmethod
def get_workers():
    return dbManager.build_fetch_query('workers',
                                       table_columns=['image_url', 'worker_description'])
```

מביא את המידע על העובדים, מתורגם לשאילתא

SELECT image_url, worker_description FROM workers

```
@staticmethod
def get_user(user_email, password=None):
    conditions = [f'email="{user_email}"']
    if password:
        conditions.append(f'password="{password}"')
    return dbManager.build_fetch_query('users', conditions=conditions)
```

מביא מידע על המשתמש ע"י האימייל, והסיסמא במידה ונתנה. מתורגם לשאילתות

- אם נתנה סיסמא – SELECT * FROM users WHERE email="{user_email}" AND password="{password}"
- אם לא נתנה סיסמא – SELECT * FROM users WHERE email="{user_email}"

```
@staticmethod
def get_tips():
    return [tip_data.tip for tip_data in
            dbManager.build_fetch_query('tips', table_columns=['tip'])]
```

מביא את המידע על הטיפים של ספי הספר. מתורגם לשאילתא

SELECT tip FROM tips

```
@staticmethod
def get_schedules_by_date(min_date, max_date):
    return {date_.schedule_time for date_ in
            dbManager.build_fetch_query('schedule', ['schedule_time'],
                                        conditions=[f'schedule_time>="{min_date}"',
                                                    f'schedule_time<"{max_date}"'])}

@staticmethod
def get_schedules_by_email(email):
    return [date_ for date_ in
            dbManager.build_fetch_query('schedule', ['schedule_time', 'schedule_type'],
                                        conditions=[f'email="{email}"'])]
```

מביא מידע על התורים שנקבעו במספרה, התאריכים משמש את השרת ע"מ לשלוח לבקשה בJS אילו תורים פנויים יש על מנת שיוכל להציג אותם למשתמש בעת קביעת התור

השני משמש לדף של צפיה בתורים של משתמש רשום, אשר מביא את כל התורים שלו ע"פ האימייל

מתורגמים לשאילתות:

- SELECT schedule_time FROM schedule where schedule_time>="{min_date}" AND schedule_time<"{max_date}"
- SELECT schedule_time, schedule_type FROM schedule WHERE email="{email}"

```
@staticmethod
def get_recommendations(limit=10):
    return [recommendation_data.recommendation for recommendation_data in
            dbManager.build_fetch_query('recommendations',
                                        table_columns=['recommendation'],
                                        order_column='created_date',
                                        order_type='DESC', limit=limit)]
```

מביא מידע על ההמלצות הכי עדכניות על ספי הספר, ככמות הlimit שהתקבל בתור ארגומנט. בדיפולט זה כ-10 המלצות. מתורגם לשאילתא

SELECT recommendation FROM recommendations ORDER BY created_date DESC LIMIT="{limit}"

```python
@staticmethod
def get_products(min_price, max_price, order_type=None, order_column=None):
    conditions = []
    if min_price:
        conditions.append(f'price >= {min_price}')
    if max_price:
        conditions.append(f'{max_price} >= price')
    return dbManager.build_fetch_query('products', order_column=order_column,
                                       order_type=order_type, conditions=conditions)

@staticmethod
def get_products_by_ids(product_ids):
    condition = f'product_id IN ({",".join(set(product_ids.split(",")))})'
    return dbManager.build_fetch_query('products', conditions=[condition])
```

מביאות מידע על המוצרים. השאילתא הראשונה מביאה מידע על המוצרים, תוך כדי אפשרויות של סינון לפי מחירי המוצרים. השאילתא משמשת את המשתמש בדף של קניית מוצרים בה יוכל המשתמש לסנן מוצרים כרצונו, בנוסף לכיצד יוצגו המוצרים (מהיקר לזול, מהזול ליקר, מהנמכר לפחות נמכר וכן הלאה.

השאילתא השנייה מביאה מידע על המוצרים ע"פ ה ID שלה.

השאילתות מתורגמות ל:

- אם ניתנו עם max_price=200, min_price=100, order_type=price, order_type=DESC
SELECT * from products WHERE price >= 100 AND price <= 200 ORDER BY price .DESC
שאילתות דומות יכולות להוויצר לפי הארגומנטים שניתנו, ניתן לראות לכך דוגמאות בקוד.
- SELECT * FROM products WHERE product_id IN (id1, id2, …)

```python
@staticmethod
def get_images_urls():
    return [photo.image_url for photo in
            dbManager.build_fetch_query('example_photos',
                                        table_columns=['image_url'])]
```

מביא את המידע על התמונות לדוגמא של תספורות שמוצגות בדף הבית של ספי הספר. השאילתא מתורגמת ל:

SELECT image_url FROM example_photos

```python
@staticmethod
def delete_item_from_cart(user_email, product_id):
    # Because product ID is not unique, need to select one row whom matches the given product ID to delete
    # Need to do 2 queries to DB because some SQL server dont support inner SELECT inside a DELETE statement.
    product_id_list = dbManager.build_fetch_query('cart_data', ['id'],
                                                  conditions=[f'user_email="{user_email}"',
                                                              f'product_id="{product_id}"',
                                                              'closed_session_date IS NULL'],
                                                  limit=1)
    if not product_id_list:
        raise ValueError('Could not find the expected item')
    cart_product_id = product_id_list[0].id
    query = f'''
    DELETE FROM cart_data
    WHERE id={cart_product_id}
    '''
    return dbManager.commit(query)
```

בעת מחיקה של חפץ מהעגלה, מביאים את המוצר שאותו ביקש המשתמש למחוק (לאחר מכן מבוצעת מחיקה – בשאילתות מסוג מחיקה נרחיב עליה). הצורך בלחלק ל2 שאילתות רשום בהערה מתחת לשם הפונקציה. מתורגם לשאילתה:

SELECT id FROM cart_data WHERE user_email="{user_email}" AND product_id="{product_id}" AND closed_session_date IS NULL LIMIT 1

מביאים רק את אלו שהתאריך הוא NULL כיוון שאלו המוצרים שקיימים עבור סשן עגלה פתוח. כאשר הסשן עגלה נסגר אז התאריך חתום עם ערך של תאריך ביצוע העסקה ולכן נרצה רק לסנן את אלו מהסשן הפתוח.

```python
@staticmethod
def get_images_urls():
    return [photo.image_url for photo in
            dbManager.build_fetch_query('before_after_photos',
                                        table_columns=['image_url'])]
```

מביא מידע על כל התמונות של לפני ואחרי של ספי הספר. מתורגם לשאילתה:

SELECT image_url FROM before_after_photos

```
@staticmethod
def get_user_active_cart(user_email):
    query = f'''
    SELECT cart_data.product_id, products.product_name, products.price, products.image_url
    FROM cart_data
    INNER JOIN products
    ON cart_data.product_id=products.product_id AND user_email="{user_email}"
    AND closed_session_date IS NULL;
    '''
    return dbManager.fetch(query)
```

מביא מידע על העגלה הפעילה של המשתמש, ומחבר אליהם את המידע שצריך על המוצרים שצריך ע"מ להציג לו את עגלתו הנוכחית. השאילתה לא משתמשת בפונקציה הגנרית כיוון שהיא קצת יוצאת דופן בעקבות הJOIN בשאילתה.

שאילתות מסוג INSERT

```
CONTACT_COLS = ['phone', 'email', 'contact_name', 'contact_description']

def add_contact_request(self, phone, email, name, description):
    return dbManager.build_insert_query('contact_us',
                                        self.CONTACT_COLS,
                                        [[f'"{phone}"', f'"{email}"', f'"{name}"', f'"{description}"']])
```

הכנסת בקשה חדשה ליצור קשר. מתורגמת לשאילתה:

INSERT INTO contact_us (phone, email, contact_name, contact_description) VALUES ("{phone}", "{ email}", "{ name}", {" description"})

```
@staticmethod
def add_schedule(service_type, phone, schedule_time, email):
    return dbManager.build_insert_query('schedule',
                                        ['schedule_time', 'phone', 'schedule_type', 'email'],
                                        [[f'"{schedule_time}"', f'"{phone}"', f'"{service_type}"', f'"{email}"']])

@staticmethod
```

הכנסת תור חדש לבסיס הנתונים. מתורגמת לשאילתה:

INSERT INTO schedule (schedule_time, phone, schedule_type, email) VALUES ("{schedule_time}", "{phone"}, "{service_type"}, "{email}")

```
@staticmethod
def add_transaction(cc_num, user_id, cvv, total, exp_date):
    return dbManager.build_insert_query('transactions',
                                        ['cc_number', 'user_id', 'cvv', 'total', 'expDate'],
                                        [[f'"{cc_num}"', user_id, cvv, total, f'"{exp_date}"']])
```

הכנסת עסקת אשראי חדשה לבסיס הנתונים, מתורגמת לשאילתה:

INSERT INTO transactions (cc_number, user_id, cvv, total, expDate) VALUES ("{cc_num}", "{user_id}", "{cvv}", {"total"}, "{exp_date}")

```
def insert_user(self, user_email, password, first_name, last_name):
    return dbManager.build_insert_query('users', self.USER_COLUMNS,
                                        [[f'"{user_email}"', f'"{password}"',
                                          f'"{first_name}"', f'"{last_name}"']])
```

הכנסת משתמש חדש שנרשם לאתר של ספי הספר, מתורגמת לשאילתא

INSERT INTO users (user_email, password, first_name, last_name) VALUES ("{user_email}", "{password}", "{first_name}", "{last_name}")

```
@staticmethod
def add_product_to_cart(user_email, product_id):
    return dbManager.build_insert_query('cart_data', ['user_email', 'product_id'],
                                        [[f'"{user_email}"', f'"{product_id}"']])
```

מכניס מוצר חדש אותו בחר משתמש לעגלה שלו, מתורגמת לשאילתא

INSERT INTO cart_data (user_email, product_id) VALUES ("{user_email}", "{product_id}")

<u>שאילתות מסוג UPDATE</u>

```
@staticmethod
def close_session_payment(user_email):
    closed_cart_date = datetime.now().strftime('%Y-%m-%d %H:%M:%S')
    return dbManager.build_update_query('cart_data',
                                        {'closed_session_date': f'"{closed_cart_date}"'},
                                        conditions=[f'user_email="{user_email}"',
                                                    'closed_session_date IS NULL'])
```

עדכון על סשן עגלה פתוח שנסגר לאחר תשלום הלקוח. מתורגם לשאילתא

UPDATE cart_data SET closed_session_data="{closed_cart_date}" WHERE user_email="{user_email}" AND closed_session_date IS NULL

```
def update_user(self, user_email, password, first_name, last_name, curr_email):
    updates = {}
    for col, new_val in zip(self.USER_COLUMNS, [user_email, password, first_name, last_name]):
        if new_val:
            updates[col] = f'"{new_val}"'
    return dbManager.build_update_query('users', updates, conditions=[f'email="{curr_email}"'])
```

בדף של עדכון פרטי משתמש, מעדכן את הפרטים של המשתמש ע"פ מה שהזין

דוגמא לשאילתה בה המשתמש עדכן את השם פרטי והאימייל:

UPDATE users SET email="{cur_mail}" first_name="{first_name}" WHERE email="{curr_email}"

כיוון שהשדות בטבלה של cart_data שהן FK לemail בטבלה של users, כאשר האימייל מעודכן כאן, ה cart_data גם מתעדכן בהתאם בעזרת ההגדרה של ON UPDATE CASCADE.

<u>שאילתות מסוג DELETE</u>

```
@staticmethod
def empty_cart(user_email):
    query = f'''
    DELETE FROM cart_data
    WHERE user_email="{user_email}" AND closed_session_date IS NULL
    '''
    return dbManager.commit(query)
```

מוחק את כל המוצרים מהעגלה של המשתמש כאשר הוא מבקש לנקות את העגלה.

```
                                                    limit=1)
if not product_id_list:
    raise ValueError('Could not find the expected item')
cart_product_id = product_id_list[0].id
query = f'''
DELETE FROM cart_data
WHERE id={cart_product_id}
'''
return dbManager.commit(query)
```

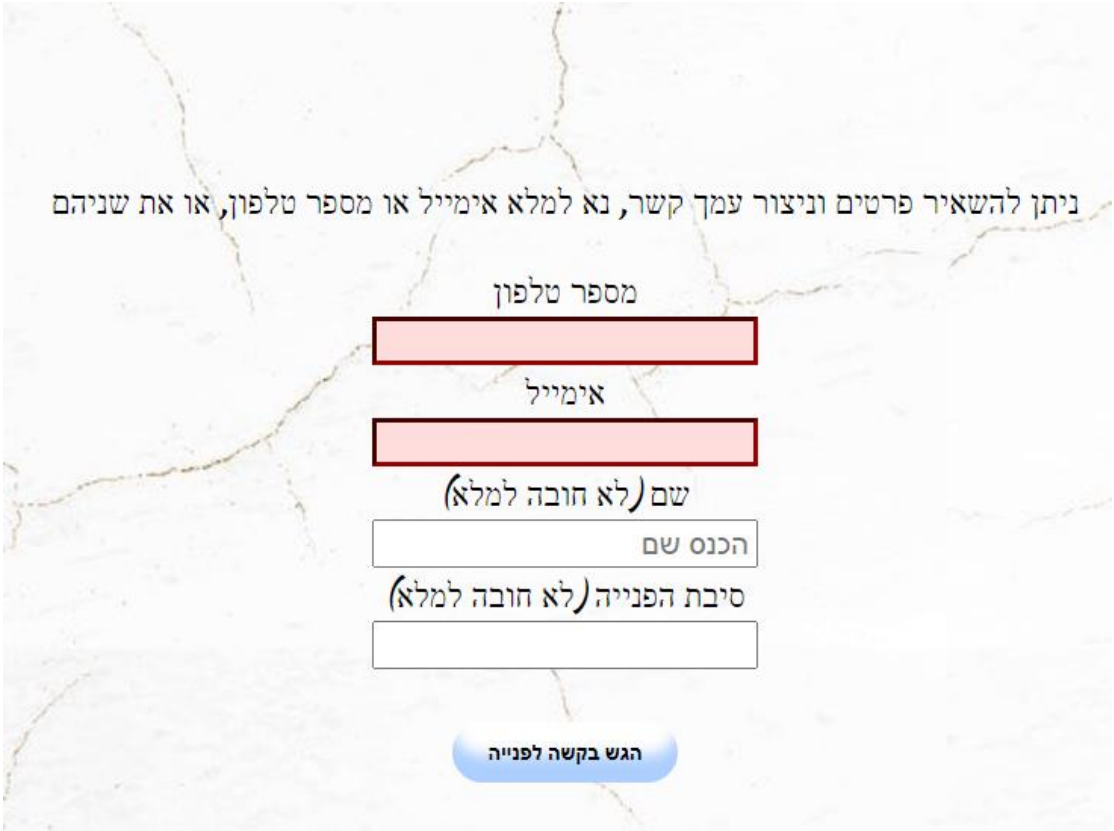
מוחק את המוצר אותו ביקש המשתמש למחוק מעגלת הקניות הנוכחית שלו.

```
@staticmethod
def cancel_schedule(schedule_time):
    query = f'''
    DELETE FROM schedule
    where schedule_time="{schedule_time}"
    '''
    dbManager.commit(query)
```

מתבצעת כאשר המשתמש מבקש לבטל תור עתידי.

5. מימוש טפסים

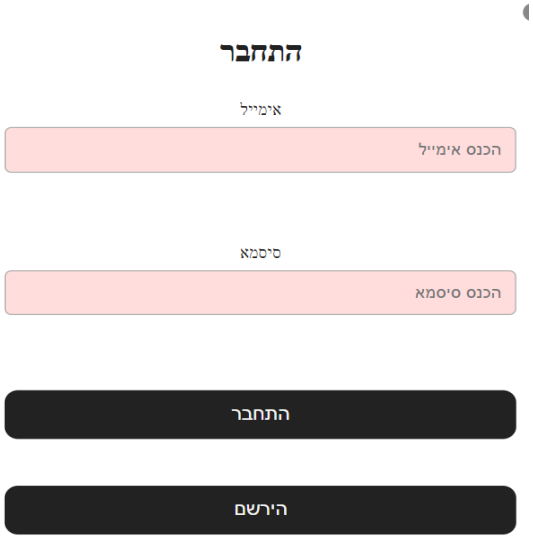
צור קשר



ניתן להגיע לעמוד זה בין אם הינך מחובר או לא, על מנת לשאול שאלות את המספרה של ספי הספר.

הACTION של טופס זה מוגדר לנתיב contact_us עם מטודת POST.

התחברות



טופס זה, כאשר כפתור "התחבר" נלחץ, שולח לשרת את הפרטים של האימייל והסיסמא, כאשר השרת מחזיר תשובה חיובית או שלילית לJS אם האימייל קיים ואכן הסיסמא תואמת.

אם הפרטים אינם נכונים מופיעה הודעה שגיאה למשתמש כי המשתמש אינו קיים או שהסיסמא אינה נכונה. אם הפרטים נכונים, מופיעה הודעה למשתמש כי הצליח להתחבר בהצלחה.

הACTION של טופס זה מוגדר לנתיב log_in עם מטודת POST.

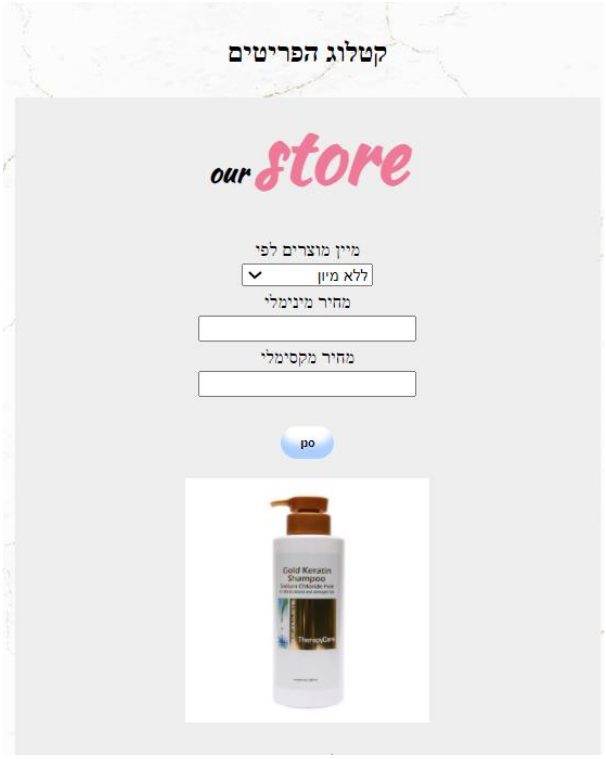
הרשמה

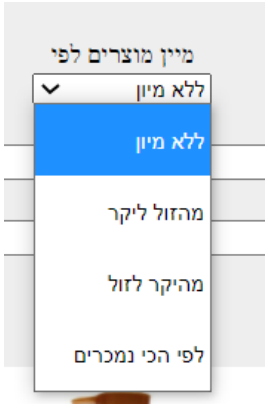
במידה ונלחץ כפתור "הירשם", נפתח הטופס הבא :

כאשר נלחץ הכפתור "הירשם", השרת מוודא כי האימייל כבר לא תפוס, במידה ותפוס מציג הודעה כי האימייל כבר בשימוש. אחרת, מודיע למשתמש כי רשם אותו ומחבר אותו למשתמש.

הACTION של טופס זה מוגדר לנתיב register עם מטודת POST.

<u>טופס סינון פריטים</u>





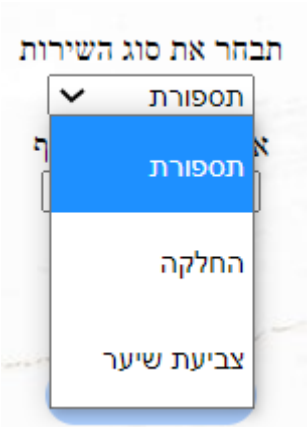
בדף של רכישת מוצרי פיתוח, ישנו טופס שיכול המשתמש למלא על מנת לסנן מוצרים, בין האפשרויות במיון מוצרים

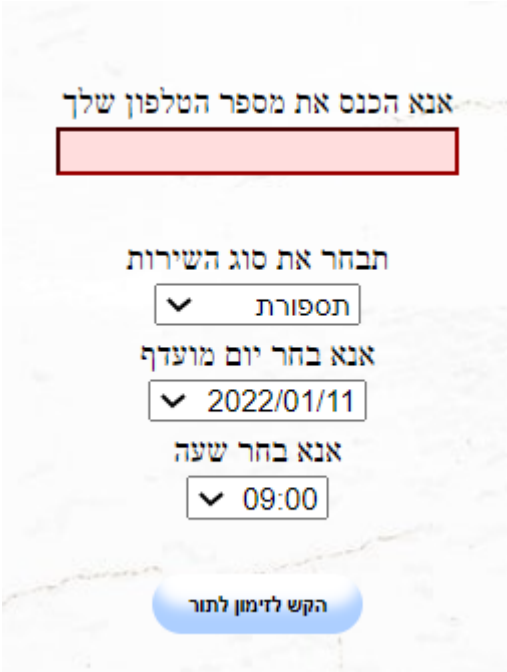
בנוסף לסינון של מחירים ממקסימלי למינימלי.

רק משתמש מחובר יכול גם להוסיף פריטים לעגלת הקניות. משתמש לא מחובר שינסה להוסיף פריטים לעגלת הקניות יקבל הודעה כי עליו להירשם לפני שיוכל לקנות באתר.

הACTION של טופס זה מוגדר לנתיב shop_items עם מטודת GET.

טופס קביעת תור

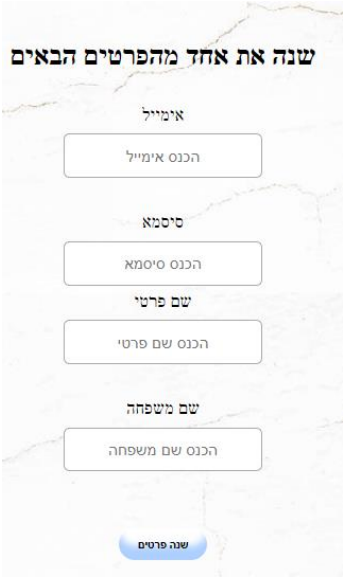




טופס אותו ניתן למלא על מנת לקבוע תור. אין צורך להיות מחובר למשתמש על מנת לקבוע תור. כאשר השעות והתאריכים שמוצגים לא כוללים ימי שבת, או שעות שאינן חלק משעות הפעילות, כולל סינון של התאריכים של התורים שכבר נלקחו

הACTION של טופס זה מוגדר לנתיב schedule עם מטודת POST.

טופס שינוי פרטי משתמש



טופס אותו יכול משתמש למלא כאשר הוא מעוניין לעדכן אחד או יותר מן הפרטים על המשתמש. במידה וילחץ על שנה פרטים ללא מילוי של שום שדה, תוקפץ לו הודעה כי אין מה לשנות.

הACTION של טופס זה מוגדר לנתיב change_details עם מטודת POST.

6. מימוש פונקציונאליות ועיבוד מידע

התחברות

כאשר המשתמש לוחץ על "התחבר", ישנן ולידציות שמוודאות שכל התאים אכן תואמים לציפיות (אימייל תקין, סיסימא) ולאחר מכן מבוצעת הפניה לשרת.

השרת בודק את תקינות הנתונים ע"י ביצוע שאילתא לבדוק האם קיימת רשומה בטבלת המשתמשים בבסיס הנתונים בה קיים המייל הזה וזוהי סיסמתו.

במידה וכן – הפרטים אכן נכונים והמשתמש מתחבר לאתר ומקבל הודעת פופ אפ כי התחבר בהצלחה.

במידה ולא – מקבל הודעה כי אחד מהפרטים שגויים. לא מדווח אם הבעיה בסיסמא או באימייל מטעמי אבטחה.

הרשמה

כאשר המשתמש לוחץ על "הירשם", ישנן ולידציות שמוודאות שכל התאים אכן תואמים לציפיות (אימייל תקין, סיסמא, שם פרטי ומשפחה שמורכבות רק מאותיות באנגלית או בעברית וכו').

לאחר הולידציות כי כל התאים תקינים, הטופס מועבר לשרת, שבודק אם קיים כבר משתמש בבסיס הנתונים עם האימייל הנ"ל.

- במידה וכן – מוקפצת הודעה כי האימייל כבר תפוס ונא לנסות עם אימייל אחר
- במידה ולא – מוקפצת הודעה כי ההרשמה התבצעה בהצלחה, ומחבר את הלקוח למשתמש החדש שלו.

הוספת פריט לעגלה

כאשר הגולש באתר מנסה להוסיף פריט לאתר – מתבצעת ולידציה כי הוא אכן מחובר למשתמש – במידה ולא, מוקפצת לו הודעה כי עליו להתחבר למשתמש על מנת לקנות באתר.

לאחר מכן, קוד הJS שולח בקשת POST לשרת בנתיב cart/add עם המספר המזהה של המוצר אותו הלקוח ביקש להוסיף.

השרת מוסיף רשומה לבסיס הנתונים המקשרת את המוצר לאימייל, ומחזיר תשובה לJS כי ההוספה התבצעה בהצלחה. לאחר מכן מוקפץ חלון ללקוח כי המוצר התווסף בהצלחה, והעמוד נטען מחדש כאשר העגלה שלו מעודכנת לאחר רכישת הפריט

מחיקת פריט מהעגלה

כאשר הגולש מבקש להסיר פריט (חייב להיות מחובר, כי אחרת לא תהיה לו עגלה), אז קוד הJS שולח בקשת POST לשרת עם הID של המוצר שנתבקשנו להסיר. השרת מוציא מהטבלה של בסיס הנתונים של העגלות שורה אחת שבה קיים המוצר, ולוקח את המזהה הייחודי של השורה ומבצע איתה בקשת DELETE.

צפייה בתורים

כאשר הלקוח מסתכל על תורים, השרת ניגש לבסיס הנתונים ומביא את כל התורים לפי האימייל. מבצע חלוקה ל2 קבוצות, אחת לתאריכים עד היום, ואחת לתאריכים עתידיים, ומציג למשתמש, תוך כדי אפשרות לביטול כל אחד מהתורים העתידיים. כאשר הלקוח מבקש לבטל תור עתידי, קוד הJS שולח בקשת POST עם התאריך אותו צריך להסיר, והשרת מסיר מהבסיס נתונים את התור הנ"ל, והוא נהיה שוב נגיש לבחירה.

שינוי פרטי משתמש

הלקוח יכול לשנות את פרטי המשתמש, וכמו שהוסבר בשאילתות, במידה והלקוח מבקש לשנות פרטים, מבוצעת פניה מJS לשרת עם כל הבקשות לשינויים, כאשר העדכון מתבצע גם עבור טבלאות אחרות הנשענות על טבלת המשתמשים (כיוון שהן מוגדרות ע"י ON UPDATE CASCADE).

קביעת תור

ספי הספר היא מספרה בה ניתן לקבוע תור עד 30 ימים קדימה לכל היותר, ולא בימי שבת.

בנוסף, לא ניתן לקבוע תור באתר מהיום להיום, שכן זה יכול לבלבל את ספי הספר והוא לא יוכל לארגן את הבאים למספרה כמו שרוצה.

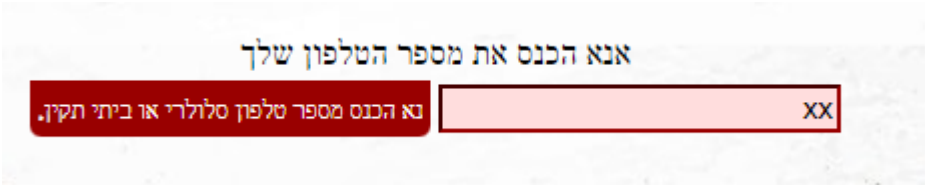
לכן, כאשר לקוח מגיע לבקשה של קביעת תור

- השרת מחשב את ה30 ימים הבאים, ומוריד את ימי שבת
- קוד הHTML מקבל את הרשימה הדינאמית של הימים אותם חישב השרת, ושולח בקשה לשרת על מנת שיתן לו את השעות הפנויות ביום שנבחר ע"י הלקוח.

ולידציות בJS

בחרנו לממש את הולידציות על ידי הודעת שגיאה אינטראקטיבית למשתמש, כאשר כל פעם בה המשתמש מכניס קלט מוצגת לו הודעה עם הסבר מה לא נכון בתא עד אשר התא מתמלא בערך ולידי (ע"י הוספת EVENT LISTENER על פעולת INPUT)

לדוגמא



הולידציה הנ"ל נותנת למשתמש חוויה יותר נעימה באתר, כאשר הגשת הטופס תהיה בהכרח (חוץ ממקרה מאוד ספציפי, של צור קשר) ולידית, במקום לתסכל את המשתמש שכל פעם יצטרך להגיש את הטופס ולראות מה לא תקין ולתקן.