Project Requirement

Beat Me

CS 157A - TEAM 1
Kun He, Tien Pham, Han Kang

Dec 12th, 2019

CS 157A - [Section 01]

1. Project Overview

1.1. Background

- The number of e-commerce shoppers is growing every year. Often, they have a difficult time to find the best price for what they are looking for. For example, on Amazon Prime Day, because lots of products are on sale at the same time, the shopper may have a hard time to figure out which products provide the best price. Also, the cost of the product is continually changing, and the list of the item looks very similar.

1.2 Customer or Market Needs

- According to *Figure 1.* by Statista, the number of e-commerce shoppers in the United States is growing. Therefore, the online deal tracker would help them to find the best price for the product. Currently, there is a website called *CamelCamelCamel,* which allows the online shopper to keep track of the item price, yet it is only for Amazon. *Beat Me* keeps track of the item price over verified online retailers so that the shopper is more likely to find the best price for the specific item.

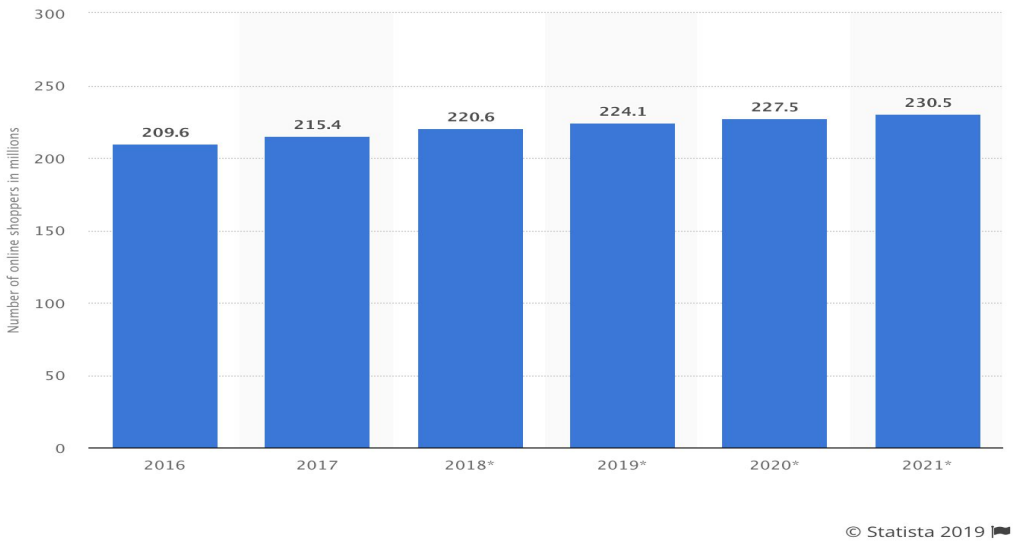


Figure 1. Number of digital shoppers in the United States from 2016 to 2021 (in millions)

1.3 Stakeholder Profile

1.3.1 Clients

- Anyone can use *Beat Me* to find the best price on the online retailer's website. *Beat Me* will guarantee the user's convenience because *Beat Me* will notify the user when the price cheaper than usual so that the user does not need to keep checking the price of the product.

1.3.2 Investors

- The small online retailer could have more benefits than big online retailers such as Amazon or eBay because Beat Me will provide advertising effects.

2. System Environment

2.1 A Structure Diagram of The System

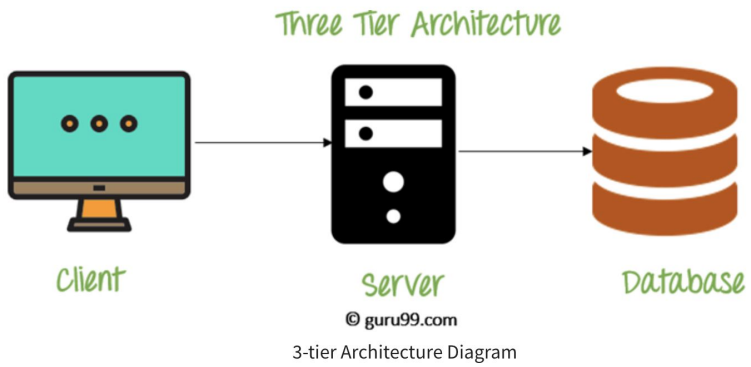


Figure 2. 3-Tier Architecture

2.2 System Requirement

- Front-end(Client): React JS, HTML, CSS, Javascript
- Back-end: Flask, Python
- Server: Apache
- Database (RDBMS - Database): MySQL 8.0
- Software: Git, Github, Taiga, Google Drive.

3. Functional requirement

3.1 Performance

- To using a Smartphone or PC, the client can access the *Beat Me* website and perform price tracking on certain products.

3.2 Functionality/Features

- Supports popular retail sites such as Amazon, Bestbuy, Walmart, etc by using their provided API or web crawling.
- Allow users to track the price of a specific product on a supported retailer site.
- Notifies users through email when the price of a product drops below the user's desired price.

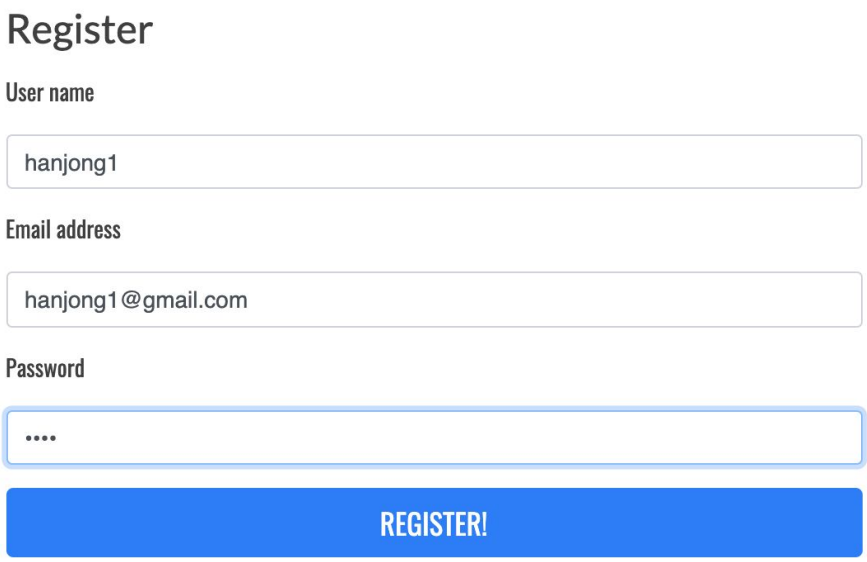
Specific Functions:

User registration

- The user shall be able to register an account with a username, email, and password
- The system will validate the user's input and check for conflicts in the database before adding the user
- Backend

```
pwd_hash = bcrypt.generate_password_hash(
    args['password'].encode('utf-8'))

try:
    sql = '''
    INSERT INTO user (username, password)
    VALUES (%s, %s)
    '''
    cursor.execute(sql, (new_username, pwd_hash))
    new_user_id = cursor.lastrowid

    sql = '''
    INSERT INTO email (address, user_id)
    VALUES (%s, %s)
    '''
    cursor.execute(sql, (new_email, new_user_id))
    new_email_id = cursor.lastrowid

    sql = '''
    INSERT INTO user_primary_email (user_id, email_id)
    VALUES (%s, %s)
    '''
    cursor.execute(sql, (new_user_id, new_email_id))
    mysql.get_db().commit()
```

- Frontend



- MySQL Database

	id	username	password
▶	1	mikewu	BLOB
	2	mikewu2	BLOB
	3	mikewu3	BLOB
	4	mikewu4	BLOB
	5	notracking	BLOB
	6	hihihi	BLOB
	8	poweruser	BLOB
	10	powerhan	BLOB
	12	ronnie	BLOB
	14	hanjong1	BLOB

User login

- The user shall be able to log in to his/her account with the correct username/email and password combination
- The system will confirm the user's credential stored in the database before granting access to the user's product tracking page
- Backend

```
class UserLogin(Resource):
    def post(self):
        args = login_parser.parse_args()
        if len(args['userid']) > 100:
            return {'message': 'username or email should be '
                               'less than 100 characters'}, 400
        try:
            cursor = mysql.get_db().cursor()
            sql = '''
            SELECT user.id, password, username FROM user JOIN email
            ON user.id = email.user_id
            WHERE email.address = %s or user.username = %s
            '''
            cursor.execute(sql, (args['userid'], args['userid']))
            result = cursor.fetchone()
        except OperationalError as e:
            return error_resp(e)

        if not result:
            return ({'message': 'incorrect userid/password combination'},
                    409)

        user_id, password, username = result
        if bcrypt.check_password_hash(password.decode('utf-8'),
                                      args['password']):
            access_token = create_access_token(identity=user_id)
            refresh_token = create_refresh_token(identity=user_id)
            resp_body = {'message': f'username {username} '
                                    f'logged in successfully'}

            resp = jsonify(resp_body)
            set_access_cookies(resp, access_token)
            set_refresh_cookies(resp, refresh_token)
            return resp

        return ({'message': 'incorrect userid/password combination'},
                409)
```

- Frontend

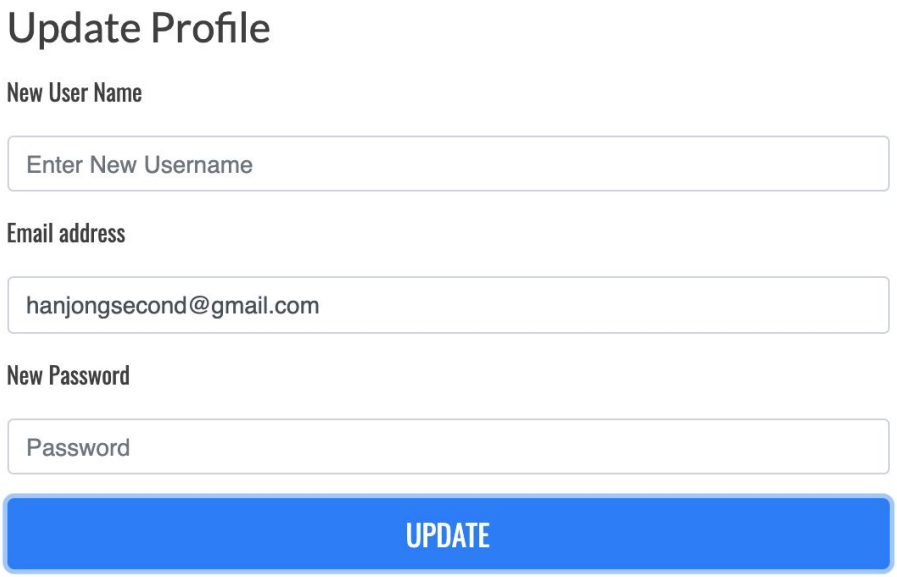
Update user information

- The user shall be able to update their email for receiving notification
- The system will update the email address of the user in the database
- Backend

```
def change_username(db, user_id, new_username):
    """
    change username in DB
    :param db:
    :param user_id:
    :param new_username:
    :return: Response Object
    """

    err = validate_username(new_username)
    if err:
        return make_response({'message': err}, 400)

    cursor = db.cursor()
    exist = username_exists(cursor, new_username)
    if exist:
        return make_response({'message': 'username already in use'}, 409)

    sql = '''
    UPDATE user
    SET username = %s
    WHERE id = %s
    '''
    try:
        cursor.execute(sql, (new_username, user_id))
        db.commit()
    except OperationalError as e:
        print(e)
        return abort(500)

    return make_response({'message': 'username updated',
                          'username': new_username})
```

```python
def add_email(db, user, new_email):
    """
    Add email to database for current user
    :param db:
    :param new_email:
    :param user:
    :return: Response Object
    """

    err = validate_email(new_email)
    if err:
        return make_response({'message': err}, 400)

    cursor = db.cursor()
    try:
        # Check if email already in DB
        exist = email_exists(cursor, new_email)
        if exist:
            return make_response({'message': 'email already in use'}, 409)

        sql = '''
        INSERT INTO email (address, user_id)
        VALUES (%s, %s)
        '''
        cursor.execute(sql, (new_email, user))
        db.commit()

    except OperationalError as e:
        print(e)
        return abort(500)

    return make_response({'message': 'new email added',
                          'email': new_email})
```

- Frontend



- MySQL Database

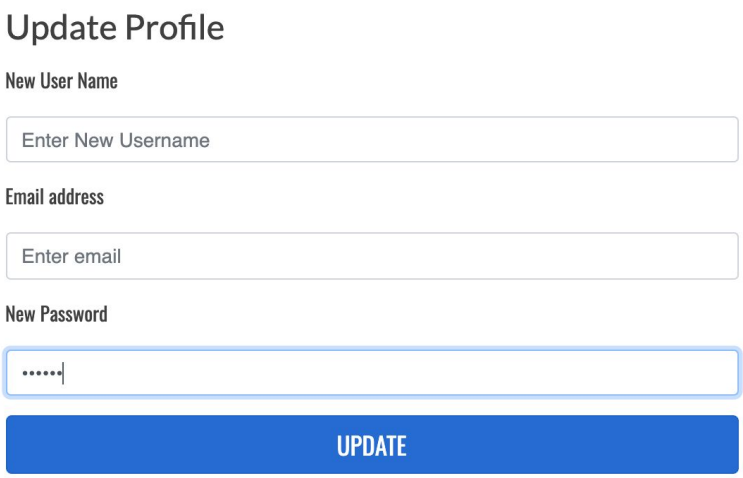
	id	address	confirmed	user_id
▶	1	mikewu@email.com	0	1
	2	mikewu2@email.com	0	2
	3	mikewu3@email.com	0	3
	4	mikewu4@email.com	0	4
	5	notracking@email.com	0	5
	6	helloworld@mail.com	0	1
	8	hi@gmail.com	0	6
	10	poweruser@gmail.com	0	8
	12	po10@gmail.com	0	8
	14	po2@gmail.com	0	8
	15	po3@gmail.com	0	8
	16	hello@gmail.com	0	8
	19	powerhan@gmail.com	0	10
	20	iamawesome@gmail.com	0	1
	26	ronnie@gmail.com	0	12
	29	hanjong1@gmail.com	0	14
	30	martin10@gmail.com	0	15
	31	asdasd@gmail.com	0	16
	32	hanjongsecond@gmail.com	0	14

Update login password

- The user shall be able to update their login password by providing their current password
- Backend

```python
def change_password(db, user_id, new_pwd):
    """
    Change user's current password
    :param db:
    :param user_id:
    :param new_pwd:
    :return: Response Object
    """

    cursor = db.cursor()
    sql = '''
    UPDATE user
    SET password = %s
    WHERE id = %s
    '''
    pwd_hash = bcrypt.generate_password_hash(new_pwd.encode('utf-8'))
    try:
        cursor.execute(sql, (pwd_hash, user_id))
        db.commit()
    except OperationalError as e:
        print(e)
        return abort(500)

    return make_response({'message': 'password updated'})
```

- Frontend



- MySQL Database: We cannot show the actual change because we hashed password and salted when we saved into the database table.

id	username	password
1	mikewu	BLOB
2	mikewu2	BLOB
3	mikewu3	BLOB
4	mikewu4	BLOB
5	notracking	BLOB
6	hihihi	BLOB
8	poweruser	BLOB
10	powerhan	BLOB
12	ronnie	BLOB
14	hanjong1	BLOB

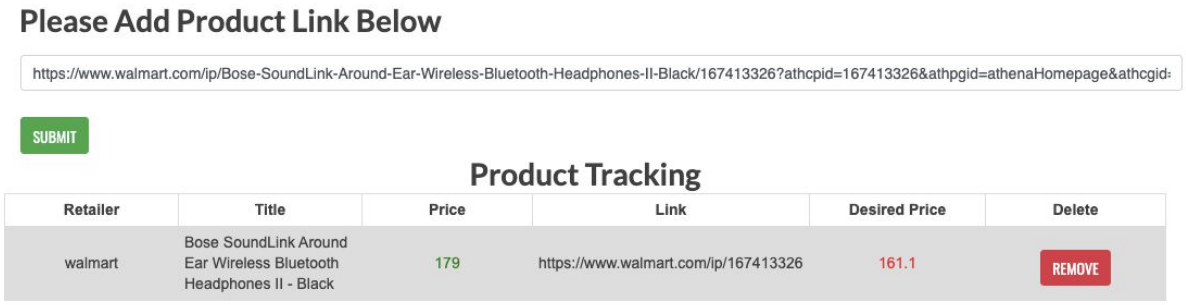
Adding product for tracking

- The user shall be able to paste the link of the product page (e.g., Amazon) and set the desired price that they are willing to pay.
- The system will add the product link to the database as well as any related items.
- Backend

```
# check if product already exists and being tracked
sql = '''
SELECT user_id, product_id, retailer
FROM product_tracked_by_user
WHERE product_id=%s and user_id=%s and retailer=%s
'''
cursor.execute(sql, (product_id, current_user_id, retailer))
if cursor.fetchone():
    return {'message': 'Product already tracked'}

# check if product already stored in DB
sql = '''
SELECT price, name, url FROM product
WHERE product_id=%s and retailer=%s'''
cursor.execute(sql, (product_id, retailer))
result = cursor.fetchone()
```


```
# check if product already exists and being tracked
sql = '''
SELECT user_id, product_id, retailer
FROM product_tracked_by_user
WHERE product_id=%s and user_id=%s and retailer=%s
'''
cursor.execute(sql, (product_id, current_user_id, retailer))
if cursor.fetchone():
    return {'message': 'Product already tracked'}

# check if product already stored in DB
sql = '''
SELECT price, name, url FROM product
WHERE product_id=%s and retailer=%s'''
cursor.execute(sql, (product_id, retailer))
result = cursor.fetchone()
```

- Frontend



- MySQL Database

name	url	price	product_id	retailer
Harry Potter and the Deathly Hallows (Book 7)	https://www.amazon.com/dp/0545139708	10.49	0545139708	amazon
Remington Smooth & Silky Electric Shaver ($5 Coupon Eligible), Pink, WDF4821US	https://www.walmart.com/ip/15686794	19.52	15686794	walmart
Bose SoundLink Around Ear Wireless Bluetooth Headphones II - Black	https://www.walmart.com/ip/167413326	179	167413326	walmart

Updating the desired price of the currently tracked product

- The user shall be able to update the wanted price of particular merchandise.
- The system updates the latest desired price of the product associated with the user in the database.
- Backend

```
def update_desired_price(db_con, user, retailer, product_id, price):
    cursor = db_con.cursor()
    sql = '''
    UPDATE product_tracked_by_user
    SET desired_price = %s
    WHERE user_id = %s AND retailer = %s AND product_id = %s
    '''
    try:
        row = cursor.execute(sql, (price, user, retailer, product_id))
        if not row:
            return make_response({'message': 'Fail to update'}, 400)
        db_con.commit()
    except OperationalError as e:
        print(e)
        return abort(500)

    return make_response({'message': 'price updated',
                          'retailer': retailer,
                          'product_id': product_id,
                          'price': price})
```

- Frontend

Retailer	Title	Price	Desired Price	New Desired Price	Update
walmart	Bose SoundLink Around Ear Wireless Bluetooth Headphones II - Black	179	161.1	150	UPDATE

Retailer	Title	Price	Desired Price	New Desired Price	Update
walmart	Bose SoundLink Around Ear Wireless Bluetooth Headphones II - Black	179	150		UPDATE

- MySQL Database

product_id	retailer	user_id	desired_price
0545139708	amazon	2	1
0545139708	amazon	8	9.441
15686794	walmart	2	20
167413326	walmart	1	161.1
167413326	walmart	12	161.1
167413326	walmart	14	150

Removing a product for tracking

- The user shall be able to remove a tracking product on their list.
- The system removes the association between the user and the product, but will not remove the product from the database.
- Backend

```
def remove_tracking(db_con, user, retailer, product_id):
    cursor = db_con.cursor()
    sql = '''
    DELETE FROM product_tracked_by_user
    WHERE user_id=%s AND retailer=%s AND product_id=%s
    '''
    try:
        row = cursor.execute(sql, (user, retailer, product_id))
        if not row:
            return make_response({'message': 'product currently not tracked'},
                                 400)
        db_con.commit()
    except OperationalError as e:
        print(e)
        return abort(500)

    return make_response({'message': 'product removed from tracking',
                          'retailer': retailer,
                          'product_id': product_id})
```

- Frontend

Product Tracking

Retailer	Title	Price	Link	Desired Price	Delete
walmart	Bose SoundLink Around Ear Wireless Bluetooth Headphones II - Black	179	https://www.walmart.com/ip/167413326	150	REMOVE

- After delete button click

Product Tracking

Retailer	Title	Price	Link	Desired Price	Delete

- MySQL Database

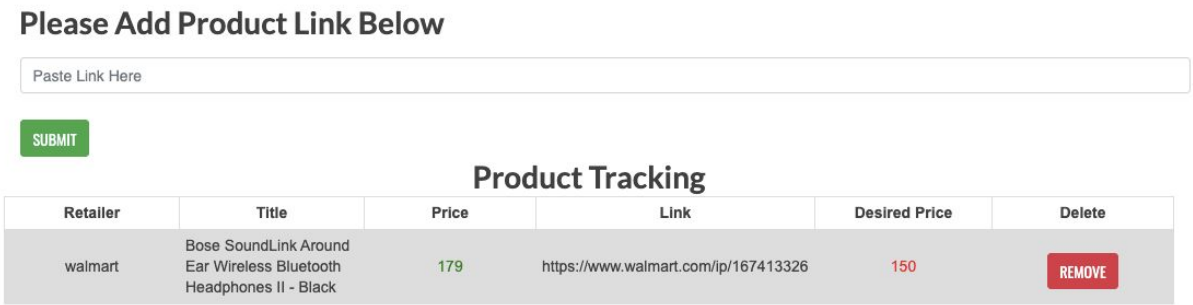
product_id	retailer	user_id	desired_price
0545139708	amazon	2	1
0545139708	amazon	8	9.441
15686794	walmart	2	20
167413326	walmart	1	161.1
167413326	walmart	12	161.1
227283391	walmart	1	18
227283391	walmart	8	17.973

Browsing tracked products

- The user should be able to browse all the products they've tracked, displaying their current price and the desired price, as well as links to the actual product.
- The system will fetch all the product information that the user is tracking from the database and display it for the user.
- Backend

```
@jwt_required
def get(self):
    current_user = get_jwt_identity()
    try:
        cursor = mysql.get_db().cursor()
        sql = '''
        SELECT product.product_id, name, url, price, desired_price,
        product.retailer
        FROM product JOIN product_tracked_by_user
        ON product.product_id = product_tracked_by_user.product_id
        and product.retailer = product_tracked_by_user.retailer
        WHERE user_id=%s
        '''
    except OperationalError as e:
        return error_resp(e)
```

- Frontend



- MySQL Database

name	url	price	product_id	retailer
Harry Potter and the Deathly Hallows (Book 7)	https://www.amazon.com/dp/0545139708	10.49	0545139708	amazon
Remington Smooth & Silky Electric Shaver ($5 Coupon Eligible), Pink, WDF4821US	https://www.walmart.com/ip/15686794	19.52	15686794	walmart
Bose SoundLink Around Ear Wireless Bluetooth Headphones II - Black	https://www.walmart.com/ip/167413326	179	167413326	walmart
Sky Rider Thunderbird Quadcopter Drone with Wi-Fi Camera, DRW389, Black	https://www.walmart.com/ip/227283391	19.97	227283391	walmart
LG 75" Class 4K UHD 2160p LED Smart TV With HDR 75UM6970PUB	https://www.walmart.com/ip/286900371	1129.99	286900371	walmart
Women's Time and Tru Lace Up Boot	https://www.walmart.com/ip/455761693	19.98	455761693	walmart
LUCID 10" Dual-Layered Gel Memory Foam Mattress, Queen	https://www.walmart.com/ip/47903235	136.99	47903235	walmart
Instant Pot LUX60 V3 6 Qt 6-in-1 Multi-Use Programmable Pressure Cooker, Slow Cooker, Rice Cooker…	https://www.walmart.com/ip/55505580	54.99	55505580	walmart
The Pioneer Woman Fiona Floral and Vintage Floral 1.5-Quart Slow Cookers, Set of 2	https://www.walmart.com/ip/679540583	34.99	679540583	walmart
Call of Duty: Modern Warfare, PlayStation 4: Get 3 Hours of 2XP with game purchase Only at Walmart	https://www.walmart.com/ip/706173372	38	706173372	walmart
Keurig K-Compact Single-Serve K-Cup Pod Coffee Maker, Black	https://www.walmart.com/ip/752022164	40	752022164	walmart
Straight Talk Samsung Galaxy S9 Smartphone	https://www.walmart.com/ip/910201860	399	910201860	walmart
Lasko Designer Series Ceramic Space Heater-Features Oscillation, Remote, and Built-in Timer, Beige	https://www.amazon.com/dp/B000N22JX6	45.99	B000N22JX6	amazon
DEWALT Titanium Drill Bit Set, Pilot Point, 21-Piece (DW1361)	https://www.amazon.com/dp/B004GIO0F8	23.01	B004GIO0F8	amazon

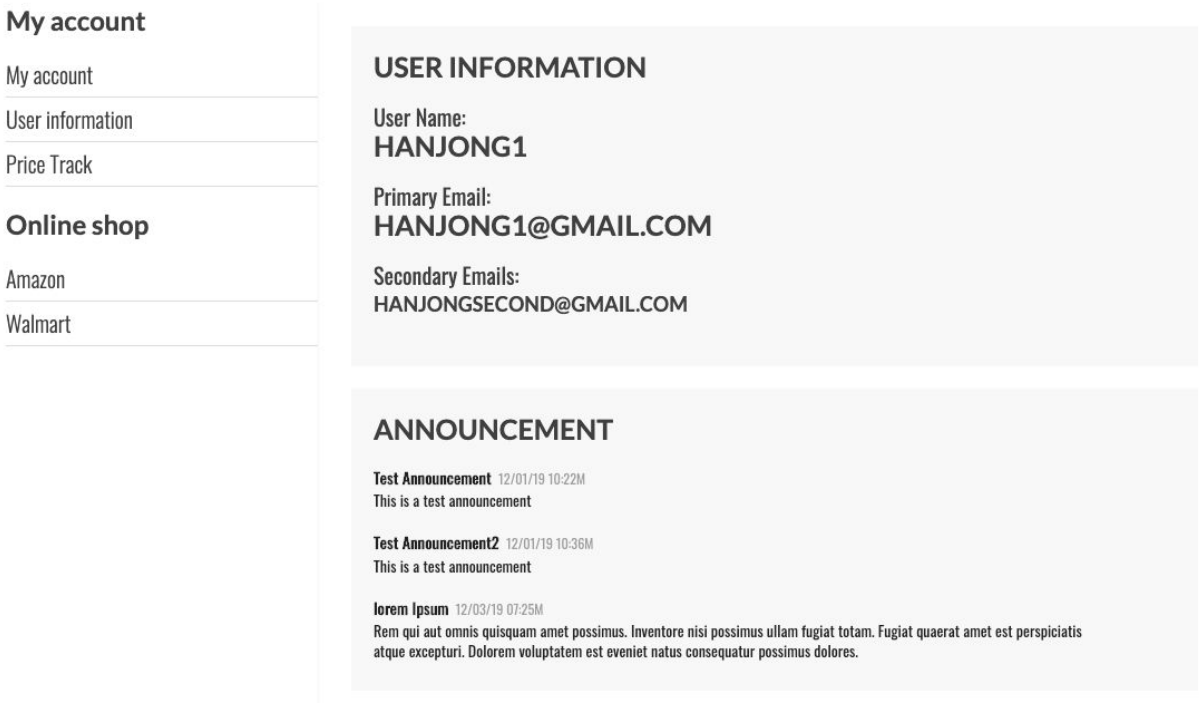
product_id	retailer	user_id	desired_price
15686794	walmart	2	20
167413326	walmart	1	161.1
167413326	walmart	12	161.1
167413326	walmart	14	150
227283391	walmart	1	18
227283391	walmart	8	17.973
455761693	walmart	1	10
55505580	walmart	1	666.66
679540583	walmart	1	31.491
706173372	walmart	1	666.66
752022164	walmart	1	36
910201860	walmart	1	359.1
B000N22JX6	amazon	1	1000
B000N22JX6	amazon	2	6666

Announcement

- The admin user can post important announcements with messages and time.
- The user will see the announcement on the main page.
- Backend

```
def get_announcement(self):
    cursor = self.db.cursor()
    sql = '''
    SELECT id, time, title, body FROM announcement
    '''
    try:
        cursor.execute(sql)
    except OperationalError as e:
        print(e)
        return abort(500)
    result = cursor.fetchall()

    ret = []
    for aid, time, title, body in result:
        ret.append({
            'id': aid,
            'time': time.strftime("%m/%d/%y %H:%M"),
            'title': title,
            'body': body
        })
    return ret
```

- Frontend



- MySQL Database

id	time	title	body	admin_id
1	2019-12-01 10:22:43	Test Announcement	This is a test announcement	3
2	2019-12-01 10:36:01	Test Announcement2	This is a test announcement	3
4	2019-12-03 07:25:18	lorem Ipsum	Rem qui aut omnis quisquam amet possimus. In…	3
NULL	NULL	NULL	NULL	NULL

Comment

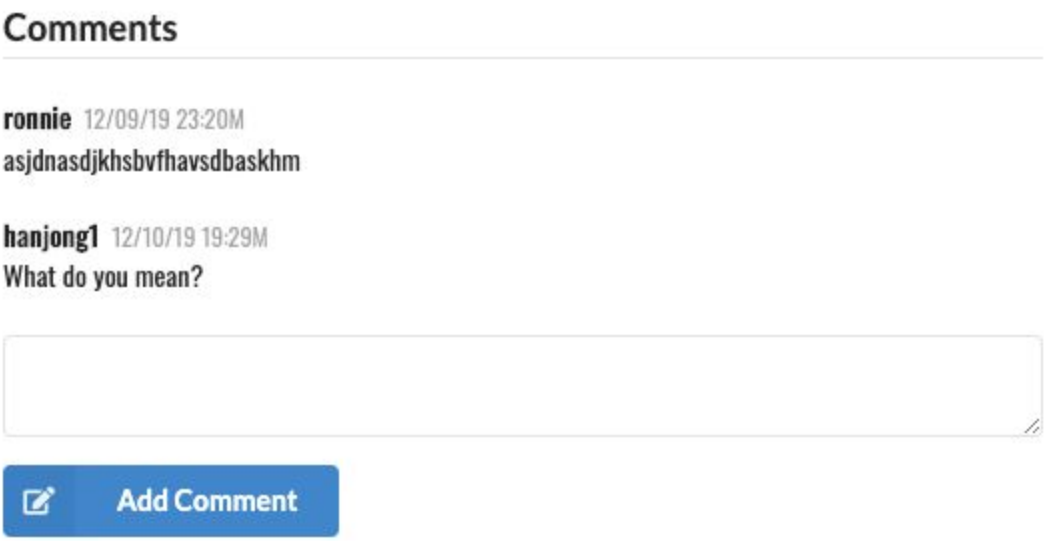
- Each user can leave comments on product's description pages.
- Once the user leaves the comment, any users can see the comment.
- Display selected product with comments
- Backend

```python
def get_comment(db, retailer, prod_id):
    cursor = db.cursor()
    sql = '''
    SELECT username, body, date FROM comment JOIN user u ON
    comment.user_id =
    u.id
    JOIN product p ON comment.product_id = p.product_id AND comment.retailer
    = p.retailer
    WHERE p.retailer = %s AND p.product_id = %s
    ORDER BY comment.date
    '''
    cursor.execute(sql, (retailer, prod_id))
    results = cursor.fetchall()
    if not results:
        return make_response({'message': 'no comments found'}, 200)
    comments = []
    for username, comment, date in results:
        comments.append({
            "username": username,
            "comment": comment,
            "date": date.strftime("%m/%d/%y %H:%M")
        })
    return comments
```

```python
def add_comment(db, user, retailer, product_id, comment):
    cursor = db.cursor()

    sql = '''
    INSERT INTO comment (body, user_id, product_id, retailer)
    VALUES (%s, %s, %s, %s)
    '''
    try:
        cursor.execute(sql, (comment, user, product_id, retailer))
        db.commit()
    except OperationalError as e:
        print(e)
        return abort(500)
    except IntegrityError as e:
        print(e)
        return error_resp('check retailer and product id', 400)
    return make_response({'message': 'comment added'}, 200)
```

- Frontend

- After ‘Add Comment’



- MySQL Database

id	body	date	user_id	product_id	retailer
1	test comment	2019-11-25 03:11:50	1	55505580	walmart
2	test comment	2019-11-25 03:35:09	1	55505580	walmart
3	test comment	2019-11-25 03:37:37	1	55505580	walmart
5	test comment	2019-11-25 03:37:59	1	55505580	walmart
7	test comment	2019-11-25 03:38:21	1	55505580	walmart
8	test comment from m2	2019-11-25 04:25:53	2	55505580	walmart
9	im brain ded	2019-12-03 05:45:22	1	55505580	walmart
10	hihihihih	2019-12-03 05:46:37	1	55505580	walmart
11	Hello	2019-12-03 09:44:41	1	B005K6ZLSK	amazon
12	I don't like it	2019-12-03 20:11:14	1	0545139708	amazon
13	Me too	2019-12-03 22:56:11	8	0545139708	amazon
14	I don't like this	2019-12-06 00:58:35	1	227283391	walmart
15	ME too	2019-12-06 19:26:31	8	227283391	walmart
16	I need this	2019-12-09 20:44:20	2	15686794	walmart
17	asjdnasdjkhsbvfhavs…	2019-12-09 23:20:17	12	167413326	walmart
18	What do you mean?	2019-12-10 19:29:33	14	167413326	walmart

4. Non-Functional issues

4.1 Security

- The user account will be password protected.
- The user's login information will be stored as hashed and salted in the database. This prevents the user's credentials from being used in case the database is breached.
- *Beat Me* website will implement with the SQL injection prevention method.

Access Control

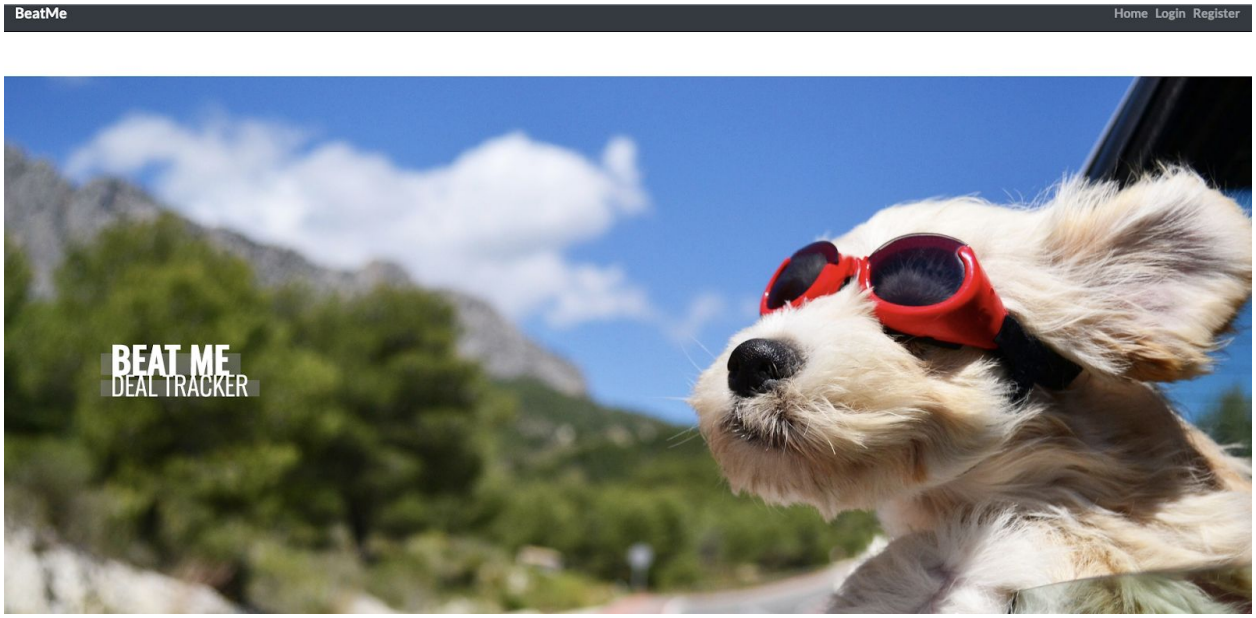
- A user will only be able to see and update his/or her product tracking list associated with their account.

Graphical User Interface (GUI)

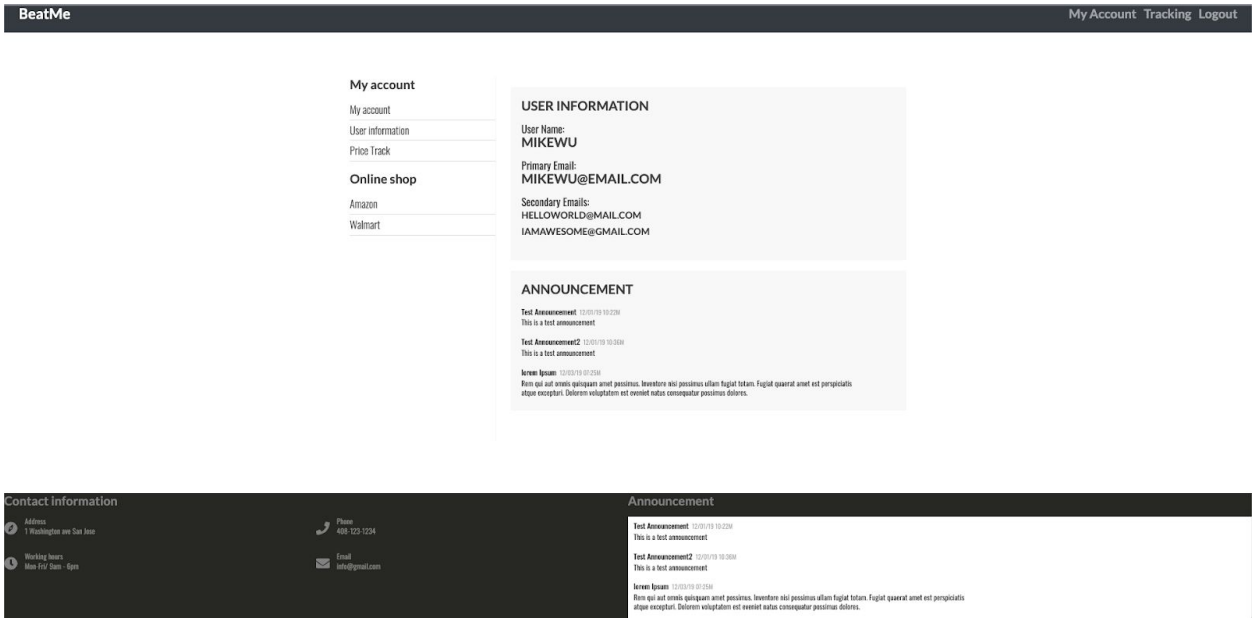
- Our GUI will aim to maximize the ease of use for the user. The user will be able to see the list of products they are tracking. It describes as columns of name, current price, desired price, and the price difference of the present and wanted price.
- The user can sort the list based on each of the columns in ascending or descending order. We will also have a field at the top of the page to enter/paste in a new product link, which will then prompt the user to enter the desired price for the product. We will have a few pre-calculated values based on the current product price allowing

ease of setting the desired price, for example, 10%, 20%, 50% off the current price.

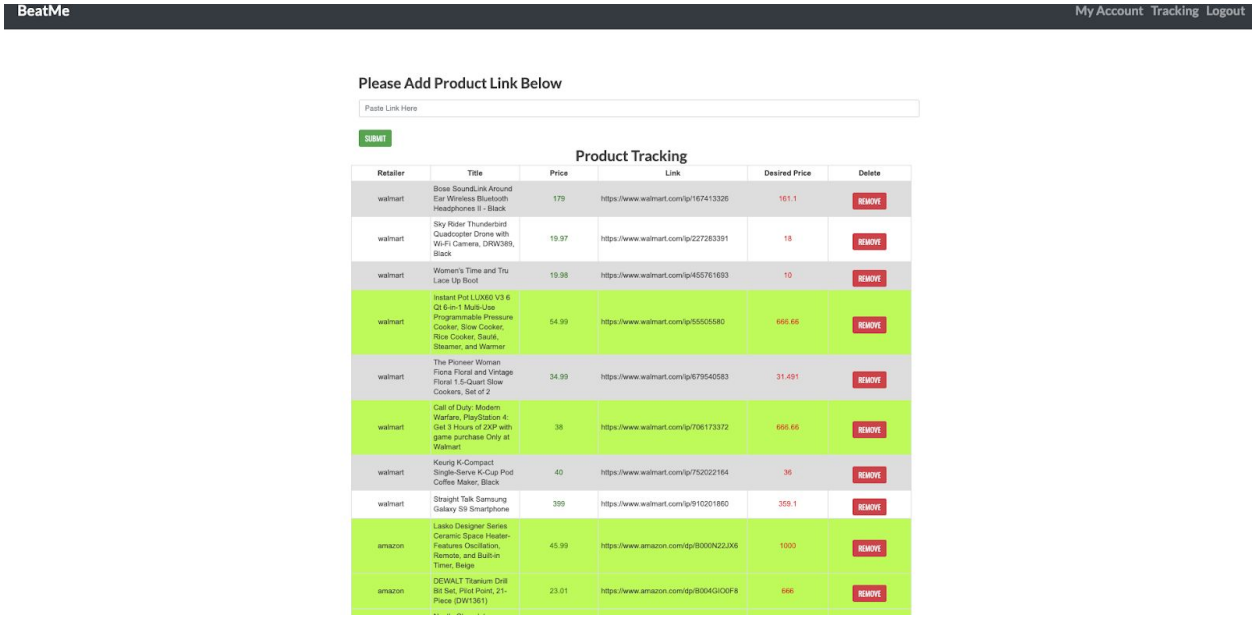
- Landing Page



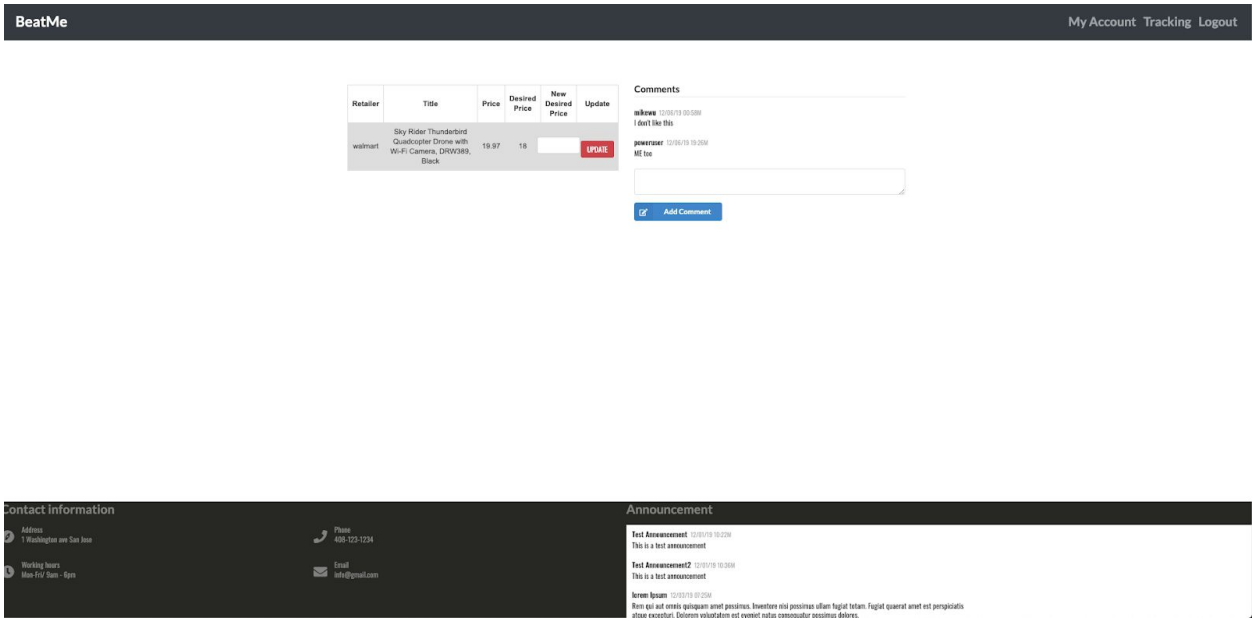
- My account page



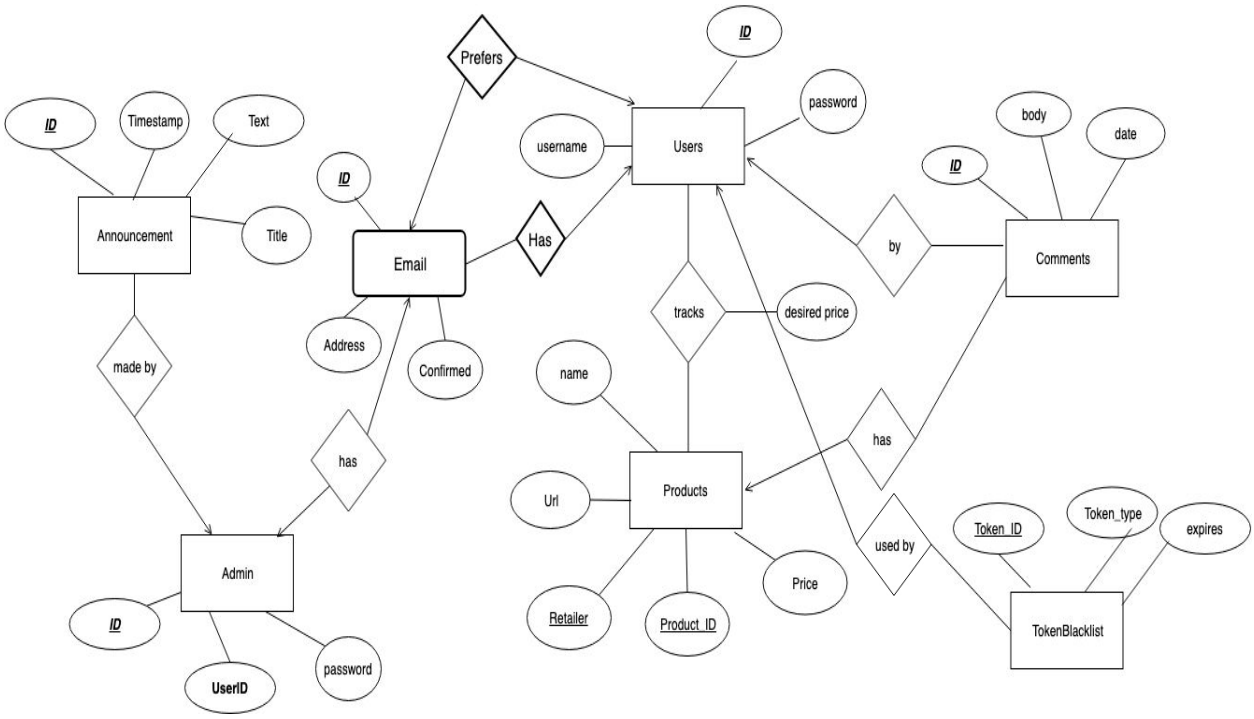
- **Product price tracking page**



- **Individual product information with comments**



5. E/R Diagram



5.1 Database Schema

- user(<u>id</u>, username, password)
- email(<u>id</u>, address, confirmed, user_id)
- user_primary_email(<u>user_id</u>, <u>email_id</u>)
- product(<u>id</u>, name, url)
- product_tracked_by_user(<u>product_id</u>, <u>user_id</u>, desired price)
- retailer(<u>id</u>, name)
- product_sold_by_retailer(<u>product_id</u>, <u>retailer_id</u>, price)
- comment(<u>id</u>, date, body, user_id, product_id)
- announcement(<u>id</u>, time, title, text, admin_id)
- token_blacklist(<u>tokenId</u>, Token_type, expires, user_id)
- admin(<u>id</u>, admin_login, password)

5.2 Database Description & Screenshot

- **User**: user table stores all users that has registered with the system with id, username, and password

	id	username	password
▶	1	vlind	BLOB
	2	milo.conroy	BLOB
	3	gerson33	BLOB
	4	eladio09	BLOB
	5	goodwin.lacy	BLOB
	6	bradtke.trystan	BLOB
	7	wrowe	BLOB
	8	mwolff	BLOB
	9	qkoch	BLOB
	10	norval07	BLOB
	11	estanton	BLOB
	12	jade.bruen	BLOB
	13	ines.skiles	BLOB
	14	wbailey	BLOB
	15	xstrosin	BLOB
	16	chelsea72	BLOB
	17	thurman79	BLOB

- **Email** email table stores all the email registered in the system and relates many-to-one to the user table

	id	address	confirmed	user_id
▶	1	tremblay.solon@rempel.com	0	1
	2	herman.hauck@starkmayer.com	0	2
	3	bailee76@schowalter.com	0	3
	4	kuvalis.kiara@schadenkuvalis.com	1	4
	5	ondricka.daphnee@nienow.com	1	5
	6	jaskolski.giovanna@fadel.com	1	6
	7	zita09@gmail.com	0	7
	8	rosenbaum.britney@gmail.com	0	8
	9	luigi25@bartoletti.info	0	9
	10	xcummings@gmail.com	0	10
	11	schoen.brenden@yahoo.com	0	11
	12	qbernier@sanfordpouros.com	1	12
	13	brenda90@bosco.com	0	13
	14	heidenreich.jamaal@hauckratke....	1	14
	15	leuschke.mazie@hotmail.com	1	15
	16	dana41@kassulkedamore.biz	0	16
	17	geovanny72@conroy.info	0	17

- **User_primary_email:** stores the primary email that the user has set, relates email and user table

user_id	email_id
1	1
2	2
3	3
4	4
5	5
6	6
7	7
8	8
9	9
10	10
11	11
12	12
13	13
14	14
15	15
16	16

- **Product**: stores products that have been added to track in the system so far, product can be tracked by multiple users with different desired price

id	name	url
1	Aspernatur enim voluptas blanditiis laboriosam r…	http://www.hansen.com/
2	Aliquid qui provident debitis deserunt vero dolor…	http://www.little.com/
3	Nemo tempore quis rerum dolores.	http://graham.org/
4	Minima rerum odit ut similique consectetur aliquid.	http://www.langworth.com/
5	Veniam minus rerum soluta totam molestiae quia.	http://www.wilkinsonrunolfsdottir.info/
6	Perferendis neque nam harum amet ut sed assu…	http://friesengerlach.biz/
7	Consequatur aut excepturi perferendis qui.	http://www.wilkinsoncasper.com/
8	Expedita qui quam velit.	http://www.bode.org/
9	Eos debitis consequatur sed quam est perferen…	http://fisher.com/
10	Dolores omnis nulla et cum nulla.	http://www.dickinson.net/
11	Consectetur dolor qui quia numquam voluptate.	http://www.zboncak.info/
12	Sed aliquid et facere odit aliquam.	http://aufderharromaguera.org/
13	Molestiae saepe facilis nulla et.	http://gottlieb.com/
14	Perspiciatis minus velit tempore libero cumque p…	http://spencer.com/
15	Deserunt voluptatem pariatur quaerat voluptat…	http://maggiodickens.biz/
16	Itaque et quia ratione eaque assumenda.	http://kessler.com/
17	Cupiditate aut fugiat cumque.	http://www.nolanschuster.com/

- **Product_tracked_by_user**: store information about which product is tracked by which user and at what desired price

product_id	user_id	desired_price
1	1	0
2	2	4.49669
3	3	807.401
4	4	25547600
5	5	488729
6	6	0
7	7	457.68
8	8	4399740
9	9	52614.5
10	10	19284.5
11	11	32.3629
12	12	44
13	13	0.6
14	14	216.848
15	15	17416.6
16	16	2660.65

- **Retailer:** store the retailer that sells product in the product table

id	name
1	molestias
2	autem
3	ut
4	delectus
5	cupiditate
6	nemo
7	qui
8	similique
9	ipsa
10	corrupti
11	non
12	aut
13	nostrum
14	velit
15	nam
16	odit
17	aut

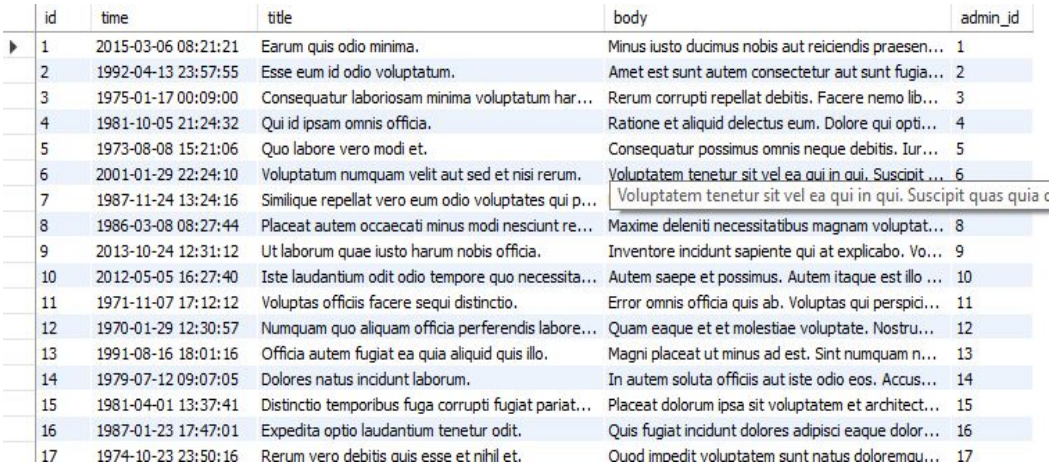
- **Product_sold_by_retailer**: store information about which product is sold by which retailer and at what price

product_id	retailer_id	price
1	1	65.6
2	2	567.218
3	3	11.467
4	4	18913.6
5	5	5.13326
6	6	0.6613
7	7	710917
8	8	322.009
9	9	0
10	10	0
11	11	13.3
12	12	1344.82
13	13	357.156
14	14	3.2
15	15	9780460
16	16	362687
17	17	0

- **Comment:** comment table relates to user and product where each user can make multiple comments on one or more product and each product can have multiple comments by one or more users

id	body	date	user_id	product_id
1	Temporibus vitae autem consectetur nesciunt. ...	1971-05-23 20:16:12	1	1
2	Non doloribus fuga consequatur nisi rem aspern...	1997-07-06 18:10:54	2	2
3	Sit non delectus consectetur pariatur et illo atqu...	2009-02-27 03:36:51	3	3
4	Et et dolorum et delectus velit accusamus fuga. ...	2001-01-09 10:37:26	4	4
5	Recusandae animi aspernatur magnam architect...	1998-06-28 17:20:36	5	5
6	Sed aut consequatur repellat fugit tempora sint...	2002-04-19 12:00:12	6	6
7	Dolorem hic fugiat amet repudiandae consequu...	1996-04-03 13:43:47	7	7
8	Dolor rerum aut et aliquid officiis quaerat. Labor...	1998-02-21 18:48:21	8	8
9	Ut corporis itaque voluptas eum. Dolores facilis ...	2019-04-14 15:02:44	9	9
10	Id consectetur occaecati fugiat animi incidunt d...	2001-06-13 10:34:25	10	10
11	Maxime et est a est explicabo doloribus. Invent...	2002-04-12 08:33:41	11	11
12	Adipisci qui itaque non totam velit. Nesciunt odi...	1986-03-04 03:41:18	12	12
13	Deserunt libero illo omnis. Labore distinctio vel r...	1989-07-14 11:41:33	13	13
14	Dolorem incidunt atque reprehenderit fuga ratio...	1995-01-28 19:50:18	14	14
15	Exercitationem magni hic repudiandae itaque la...	1984-01-17 21:50:13	15	15
16	Et nesciunt cupiditate error omnis sunt. Pariatur...	1978-09-21 04:40:27	16	16
17	Voluptatem nesciunt culpa ut aut sunt. Tempora...	1995-08-05 23:42:23	17	17

- **Announcement:** This table stores title, body text, and the current time. It identified with id. Announcement table has a relationship with admin table. The registered admin user can make announcements.



id	time	title	body	admin_id
1	2015-03-06 08:21:21	Earum quis odio minima.	Minus iusto ducimus nobis aut reiciendis praesen...	1
2	1992-04-13 23:57:55	Esse eum id odio voluptatum.	Amet est sunt autem consectetur aut sunt fugia...	2
3	1975-01-17 00:09:00	Consequatur laboriosam minima voluptatum har...	Rerum corrupti repellat debitis. Facere nemo lib...	3
4	1981-10-05 21:24:32	Qui id ipsam omnis officia.	Ratione et aliquid delectus eum. Dolore qui opti...	4
5	1973-08-08 15:21:06	Quo labore vero modi et.	Consequatur possimus omnis neque debitis. Iur...	5
6	2001-01-29 22:24:10	Voluptatum numquam velit aut sed et nisi rerum.	Voluptatem tenetur sit vel ea qui in qui. Suscipit ...	6
7	1987-11-24 13:24:16	Similique repellat vero eum odio voluptates qui p...	Voluptatem tenetur sit vel ea qui in qui. Suscipit quas quia c	
8	1986-03-08 08:27:44	Placeat autem occaecati minus modi nesciunt re...	Maxime deleniti necessitatibus magnam voluptat...	8
9	2013-10-24 12:31:12	Ut laborum quae iusto harum nobis officia.	Inventore incidunt sapiente qui at explicabo. Vo...	9
10	2012-05-05 16:27:40	Iste laudantium odit odio tempore quo necessita...	Autem saepe et possimus. Autem itaque est illo ...	10
11	1971-11-07 17:12:12	Voluptas officiis facere sequi distinctio.	Error omnis officia quis ab. Voluptas qui perspici...	11
12	1970-01-29 12:30:57	Numquam quo aliquam officia perferendis labore...	Quam eaque et et molestiae voluptate. Nostru...	12
13	1991-08-16 18:01:16	Officia autem fugiat ea quia aliquid quis illo.	Magni placeat ut minus ad est. Sint numquam n...	13
14	1979-07-12 09:07:05	Dolores natus incidunt laborum.	In autem soluta officiis aut iste odio eos. Accus...	14
15	1981-04-01 13:37:41	Distinctio temporibus fuga corrupti fugiat pariat...	Placeat dolorum ipsa sit voluptatem et architect...	15
16	1987-01-23 17:47:01	Expedita optio laudantium tenetur odit.	Quis fugiat incidunt dolores adipisci eaque dolor...	16
17	1974-10-23 23:50:16	Rerum vero debitis quis esse et nihil et.	Quod impedit voluptatem sunt natus doloremqu...	17

- **Admin:** store admin that can update announcements on the website

id	admin_login	password
1	david33	BLOB
2	mckayla65	BLOB
3	carter.chasity	BLOB
4	lebsack.greg	BLOB
5	lupe.kuhlman	BLOB
6	freddy.shields	BLOB
7	camryn75	BLOB
8	huel.obie	BLOB
9	jimmy.flatley	BLOB
10	mtillman	BLOB
11	dulce.powlowski	BLOB
12	gerardo96	BLOB
13	idaniel	BLOB
14	mia.becker	BLOB
15	olubowitz	BLOB
16	dusty26	BLOB
17	althea.lueilwitz	BLOB
18	nkutch	BLOB
19	melisa.predovic	BLOB
20	jheaney	BLOB
NULL	NULL	NULL

- **Token_blackList:** TokenBlackList table store token identification, type, and expiration date when a user logs out so it can't be used again. Each token belongs to one user in the user table

jti	token_type	expires	user_id
23c6bed5-32a4-3b16-9d88-cdf2cf9a09ab	impedit	1970-01-14 19:04:24	20
26256de1-6ea4-3a7d-97e5-a46d89158429	ex	1985-09-12 07:27:04	14
27169427-7ea1-3eb1-8bb2-93ebbbfccffd	occaecati	2017-05-24 21:55:52	2
2fb34fae-2aad-39b5-b866-6f73d14f2761	corporis	2009-12-17 07:22:30	8
44aec57a-3845-3eb7-b1f2-3cc220ba697b	adipisci	1984-09-20 13:35:20	6
4cc660d7-7ef7-3a1e-a32e-f2579432faa1	eum	2007-03-20 20:40:43	15
4d3e458b-0294-340a-97bb-f1a33f9b7fbc	voluptate	2015-03-30 20:25:53	19
6002667b-a8f7-3199-8c36-1f7dc8b94401	distinctio	2011-10-09 15:59:01	4
75553e61-1e11-33f5-ba60-7d21b9be476c	commodi	1997-10-16 03:18:52	10
7ca4711e-4250-3111-9fdf-e675eaf64dcc	nisi	2012-08-18 08:05:46	2
7f3ad6f8-53f4-31bf-b91e-76a43c693866	modi	1970-10-09 18:59:18	9
7fae25c3-9dd2-3870-90dd-f13f787ecdbd	voluptatem	2016-06-07 13:40:45	8
858a33da-e4b2-3c39-977f-8bc64a1adbe0	cumque	1987-12-11 12:54:21	10
88193c38-f5eb-3c79-bda8-55ebc2690540	quis	2015-05-14 16:03:36	18
93713e1b-12d5-3b22-a7da-e05980684d03	et	1982-04-21 20:47:25	9
a23e39cb-7dd7-3764-a802-65778cbb98ca	est	2009-06-24 13:45:40	7
a291adcf-c1f8-3066-bade-712f52da8106	quod	2013-02-12 22:40:27	1

6. Conclusion

- **Han, Kang: With this database project, I have learned why database design is important for IT industries. I am still learning process of designing database but I now have good knowledge to design proper database. Also, I learned that communication with teammates is important to get good project result. In the future, I would like to improve my database application with better design of ERD and clear relationship with each tables and attributes.**

- **Kun He: From this database project I've learned many new concepts of database design and why it is important to have a good database design. I will be using the knowledge I learned from this project such as ER diagram and normalization in future projects to design an efficient database for the system.**
- **Tien: In this project, I learned to write a real-life project functional requirements. I researched how three-tiers of architecture is applied. Knowing how to use GitHub is one of the essential tools to work with other teammates. I've learned and applied some of the front end knowledge, but I still need to level up my skill to understand the source code of my teammates. I have learned that Python is one of the great languages to use in projects compared to others. However, I didn't have enough knowledge to contribute to the logic part by using Python. After this course, I'll definitely learn how to build projects with Python. For the database, most of the information in database management class is very important to create meaningful sql tables. Being a great team player is also one of the most valuable lessons I've learned from the project. I am on my way to perfecting myself; the knowledge that I have gained from this project is necessary for my development.**

REFERENCES

DBMS Architecture: 1-Tier, 2-Tier & 3-Tier. Retrieved from https://www.guru99.com/dbms-architecture.html

Number of digital shoppers in the U.S. 2021. Retrieved from https://www.statista.com/statistics/183755/number-of-us-internet-shoppers-since-2009/