



MIDDLE EAST TECHNICAL UNIVERSITY
NORTHERN CYPRUS CAMPUS

COMPUTER ENGINEERING PROGRAM

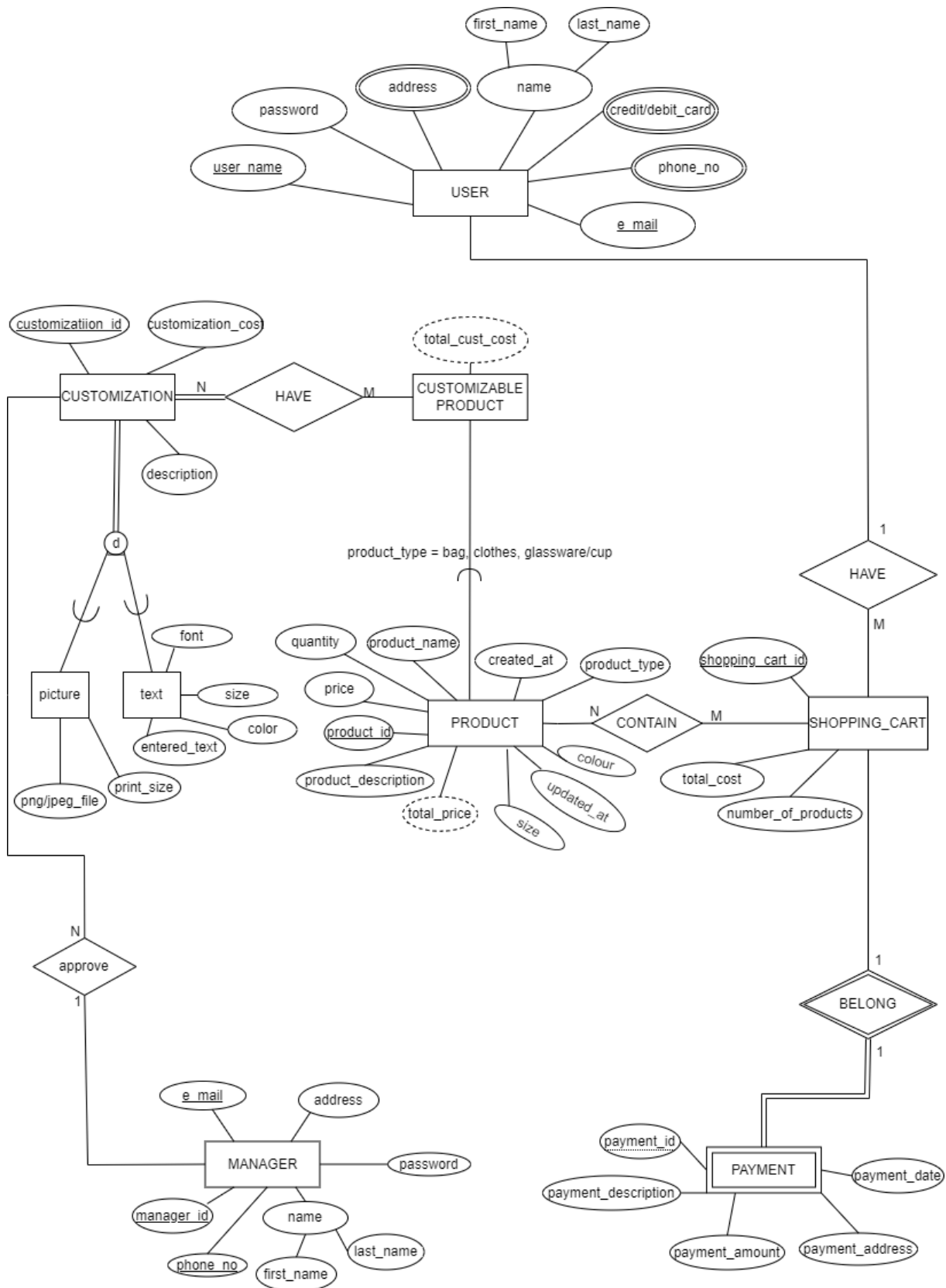
CNG 352- DATABASE MANAGEMENT SYSTEMS

PROJECT TITLE: YOUR METU STYLE

İREM ÇİLOĞLU 2385292

UMUT BAYBECE 2385201

SERGÜNEY GÜMÜŞ 2385417



Project Description

In this project, we plan to create an application for the campus store to help customers customize existing products. This application is called Your METU Style(YMS). Currently, our campus store doesn't have an online shopping webpage therefore people are not able to order products online. Also, users cannot customize their products. Therefore, the project's main goal is to help customers order products and personalize these products easily. Also, they may order products without customizing them. To order products from the Campus Store, users need to register and log in to their user accounts. Also, Campus Store managers have an admin page to receive orders and add products to the application. In this application, users can customize the products in different ways. Firstly, users can enter text with various fonts, colors, or sizes they want to print on the product. Also, they are allowed to choose the place of the text on the product. Secondly, they can upload different .png or .jpg folders to be printed on products. To sum up, users can order their customized products and purchase through our web application via online payment system.

Assumptions:

- 'user_name' is the primary key of the 'USER' entity.
- 'product_id' is the primary key of the 'PRODUCT' entity.
- 'manager_id' is the primary key of the 'MANAGER' entity.
- 'shopping_cart_id' is the primary key of the 'SHOPPING_CART' entity.
- 'admin_name' is the primary key of the 'ADMIN' entity.
- CUSTOMIZATION is a weak entity, and its primary key is user_name from the 'USER' entity.
- PAYMENT is a weak entity, and its primary keys are payment_id and shopping_card_id which is from the "SHOPPING_CART" entity.
- Admins are added to the database manually.
- Users can customize customizable products and purchase these products if the managers approve the customizations.
- Customizable products can also be purchased without any customization.
- If the manager adds a product to the database, a unique id is generated automatically for this product.
- When the manager adds a new product to the database, "created_at" and

"updated_at" are generated based on system time.

- When the manager updates an existing product in the database, "updated_at" is generated based on system time.
- "customization_cost" is derived from the text size or resolution of the picture.
- "total_cust_cost" is derived from the number of customization created for the customizable products.
- In 'PRODUCT' entity, 'total_price' = 'total_cust_cost' + 'price'.
- If there is no customization, 'total_cust_cost' equals zero.
- 'total_cost' in 'SHOPPING_CART' is sum of total_price of products in shopping cart.
- 'payment_amount' is derived from 'total_cost'.

Data Requirements

USER

Your METU Style(YMS) is for everyone who is interested in buying products from Campus Store. A visitor can see METU products on the YMS. To make a purchase, they need to register to the system. The visitor must enter their name, credit card/debit, unique phone number, unique e-mail, address, password, and unique user name to become a user. Users can order available products, customize customizable products and add them to their shopping cart. The system will hold customization history to utilize again whenever a user wants to use it. After that, they can continue with YMS's online payment system.

PRODUCT

Campus Store offers six types of products: clothes, glassware, bag, accessories, notebook, and pens. Different product types can be added later on by the manager. Each product has several attributes. They have a unique product id, price, product name, product type, product description, color, size, quantity, create and update date of the product. Also, the total cost of the products will be calculated based on price and customization price if the product is customized. Some of the products can be customized by users. Products will be added or updated by managers.

CUSTOMIZATION

Users can customize some products: clothes, bags, and glassware. They can personalize their product by adding text or pictures. Users can indicate the description's text or image location for the chosen product. The user must decide the size, color, and font of the texts. For the pictures, users should upload them in png/jpeg file format, and their resolution needs to meet the company's requirements. Also, the system will calculate the cost of every customization separately.

SHOPPING CART

Users will have a shopping cart to make an order. Users can see their products in the shopping cart and add, update or remove them. Also, the total cost will be calculated by summing up the full price of each product added to the cart. Every shopping cart has a unique shopping cart id, total cost, and the number of products derived. If there are any customized products, the shopping cart will be on hold until the manager approves the user's customization request. If manager doesn't approve it, the product is deleted from the shopping cart and user is informed. Otherwise, the shopping cart is approved and user is directed to payment page.

MANAGER

The Campus Store employees are authorized as the managers of the system. Their accounts can do certain operations on the system, such as adding and updating products and checking and approving customization of users. Managers must enter their address, a unique phone number, a unique e-mail, unique manager id, manager name, and password to register the YDS. Once they register, they can log in to the system with their e-mail and password.

PAYMENT

After approving their shopping cart on our website, Your METU Style, the users will move on to the online payment system. The payment has payment id, payment date, payment description, and payment amount taken from the shopping cart, and users should indicate their payment address. All payments are saved in the system.

Transaction Requirements

Data Entry

- Enter the details of new user. (Such as details of Umut Baybece)

- Enter the details of new products. (Such as details of T-shirt with large size and blue color)
- Enter the details of new customizable products. (Such as details of T-shirt with large size and blue color and photo link)
- Enter the details of new manager. (Such as details of İrem Çiloğlu)
- Enter the new shopping cart details. (Such as client Sergüney Gümüş adding product with id 500 in shopping cart).
- Enter the new payment details. (Such as client Sergüney Gümüş buying from shopping card with one of the recorded credit cards).
- Enter the new customization details for customizable products. (Such as client Sergüney Gümüş adding “LGBTTIQ+” text with Verdana Font, Blue, and Size 30 to a Sweatshirt).

Data Update/Deletion

- Update/delete the details of a user.
- Update/delete the details of a product by a manager.
- Update/delete the details of a manager.
- Update/delete the details of a shopping cart.
- Update/delete a product from a shopping cart.
- Update/delete the details of payment.
- Update/delete the details of customization for customizable products.

Data Queries

1. List the products whose quantity is greater than zero on the main page.
2. Group products that have the same product type as a category.
3. List the products which have selected color/s.

4. List the products which have selected size/s.
5. List the products whose name is searched.
6. List the products whose price is in the entered range.
7. List the products whose description includes entered keyword.
8. List the customizations entered by the users.
9. List the customizations approved by the managers.
10. List the products which are in the users' shopping carts.
11. List the users' payment details for each shopping cart created.
12. List the username, email, and password of users to log in.
13. List the email and password of managers to use to log in.
14. Group shopping carts for the same user.
15. List all payment transactions of a particular date.
16. List total profit of each day for a month.
17. List total profit of each month annually.
18. List online users at a moment for a manager.
19. List all shopping carts belonging a user.
20. List all products which are out of stock.
21. List all customizable products for user.
22. List all products according to updating time.
23. List products from the cheapest to the most expensive.
24. List products from the most expensive to the cheapest.
25. List addresses of user for user.
26. List all information of user for profile page.
27. List all information of manager for profile page.
28. List all products which has same entered type and are out of stock.
29. List all products with entered type according to creation time for user (the most recent products first).
30. List all products with entered type from the cheapest to the most expensive.
31. List all products with entered type from the most expensive to the cheapest.

Mapping

User (user_name, password, first_name, last_name, e_mail)

FD1: user_name → password, first_name, last_name, e_mail

FD2: e_mail → user_name, password, first_name, last_name

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Email is a candidate key and it has a functional dependency with primary key, so table violates BCNF.

User (user_name, e_mail [FK: U_email: e_mail], password, first_name, last_name)

U_email (e_mail, user_name [FK: User: user_name])

Shopping Cart (shopping_cart_id, number_of_products, total_cost,

have [FK: User: user_name])

FD1: shopping_cart_id → number_of_products, total_cost, have

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Manager (manager_id, phone_no, first_name, last_name, e_mail, password, address)

FD1 : manager_id → phone_no, first_name, last_name, e_mail, password, address

FD2 : e_mail → manager_id, phone_no, first_name, last_name, password, address

FD3 : phone_no → manager_id, first_name, last_name, e_mail, password, address

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

M_e_mail is a candidate key and it has a functional dependency with primary key, so table violates BCNF.

M_phone_no is a candidate key and it has a functional dependency with primary key, so table violates BCNF.

Manager (manager_id, phone_no, first_name, last_name, e_mail[FK:M_e_mail], password, address)

M_e_mail(e_mail, manager_id[FK: Manager: manager_id])

M_phone_no(phone_no, manager_id[FK: Manager: manager_id])

Payment (payment_id, shopping_cart_id[FK: Shopping Cart: shopping_cart_id], payment_amount, payment_address, payment_description, payment_date)

FD1: payment_id, shopping_cart_id -> payment_amount, payment_address, payment_description, payment_date

FD2: shopping_cart_id -> payment_id, payment_amount, payment_address, payment_description, payment_date

same payment_id is created again everyday, so it can't show alone.

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

P_shopping_card_id is a candidate key and it has a functional dependency with primary key, so table violates BCNF. However it is not efficient to apply BCNF we leave it in 3NF.

Payment (payment_id, shopping_cart_id[FK: Shopping_Cart: shopping_card_id], payment_amount, payment_address, payment_description, payment_date)

User Phone Number (phone_no, user_name[FK: User: user_name])

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Credit/Debit Card (credit/debit_card, user_name[FK: User: user_name])

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

User Address (address, user_name[FK: User: user_name])

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Picture (customization_id, png/jpeg_file, description, customization_cost, approve[FK: Manager: manager_id])

FD1: customization_id -> png/jpeg_file, description, customization_cost, approve

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Text (customization_id, customization_cost, description, font, size, color, entered_text, approve[FK: Manager: manager_id])

FD1: customization_id -> customization_cost, description, font, size, color, entered_text, approve[FK: Manager: manager_id]

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Product (product_id, product_description, price, quantity, product_name, created_at, updated_at, product_type, color, size, total_price)

FD1: product_id-> product_description, price, quantity, product_name, created_at, updated_at, product_type, color, size, total_price

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Customizable Product (product_id [FK: Product: product_id], total_cust_cost)

FD1: product_id -> total_cust_cost

1NF

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Have (customization_id [FK: Text, Picture: customization_id], product_id [FK: Product: product_id])

1NF-----

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Contain (shopping_cart_id[FK: Shopping Cart: shopping_cart_id], product_id[FK: Product: product_id])

1NF-----

There is no multivalued attribute. Table does not violate 1NF.

2NF

There is no composite key. Table does not violate 2NF.

3NF

Table does not violate 3NF.

BCNF

Does not violate BCNF.

Combining Relations:

1.

User (user_name, e_mail[FK: U_email: e_mail], password, first_name, last_name)

U_email (e_mail, user_name [FK: User: user_name])

We combined these tables since they could be combined without introducing redundancy.

User (user_name, password, first_name, last_name, e_mail)

2.

Manager (manager_id, phone_no, first_name, last_name, e_mail[FK:M_e_mail], password, address)

M_e_mail(e_mail,manager_id[FK: Manager: manager_id])

M_phone_no(phone_no,manager_id[FK: Manager: manager_id])

We combined these tables since they could be combined without introducing redundancy.

Manager (manager_id, phone_no, first_name, last_name, e_mail, password, address)

3.

Shopping Cart (shopping_cart_id, number_of_products, total_cost,

have[FK: User: user_name])

Payment (payment_id, shopping_cart_id[FK: Shopping Cart: shopping_cart_id], payment_amount, payment_address, payment_description, payment_date)

We combined these tables since they could be combined without introducing redundancy and they have one to one relation.

Payment_ShoppingCart(payment_id, shopping_cart_id, number_of_products, total_cost, have[FK: User: user_name], payment_amount, payment_address, payment_description, payment_date)

Finale Tables After Normalisation:

Contain (shopping_cart_id[FK: Payment_Shopping_Cart: shopping_cart_id], product_id[FK: Product: product_id])

Have (customization_id [FK: Text, Picture: customization_id], product_id [FK: Product: product_id])

Customizable Product (product_id [FK: Product: product_id], total_cust_cost)

Product (product_id, product_description, price, quantity, product_name, created_at, updated_at, product_type, color, size, total_price)

Text (customization_id, customization_cost, description, font, size, color, entered_text, approve[FK: Manager: manager_id])

Picture (customization_id, png/jpeg_file, description, customization_cost, approve[FK: Manager: manager_id])

User Address (address, user_name[FK: User: user_name])

Credit/Debit Card (credit/debit card, user_name[FK: User: user_name])

User Phone Number (phone_no, user_name[FK: User: user_name])

Payment_Shopping_Cart(shopping_cart_id, number_of_products, total_cost, have[FK: User: user_name], payment_amount, payment_address, payment_description, payment_date)

Manager (manager_id, phone_no, first_name, last_name, e_mail, password, address)

User (user_name, password, first_name, last_name, e_mail)

Workload

	T1: Listing the products				T2: Listing the customizations				T3: Listing the user information			
	I	R	U	D	I	R	U	D	I	R	U	D
contain												
credit_debit_card												
customizable_product						X						
have		X				X						
manager												
payment_shoppin_cart		X										
picture						X						
product		X				X						
text_table						X						
user_address										X		
user_phone_number										X		
user_table										X		
	T4: Listing the payment transactions				T5: List the profit				T6: Group products according to type			
	I	R	U	D	I	R	U	D	I	R	U	D
contain												
credit_debit_card												
customizable_product										X		
have												
manager												
payment_shoppin_cart		X				X						
picture												
product										X		
text_table												
user_address												
user_phone_number												
user_table												
	T7: Group shopping carts				T8: Update/delete the details of a user				T9: Update/delete the details of a product			
	I	R	U	D	I	R	U	D	I	R	U	D
contain												
credit_debit_card							X	X				
customizable_product												
have												
manager												
payment_shoppin_cart		X					X	X				
picture												
product											X	X
text_table												
user_address							X	X				
user_phone_number							X	X				
user_table		X					X	X				
	T12: Update/delete a product from the cart				T13: Update/delete the details of payment				T14: Update/delete the customization details			
	I	R	U	D	I	R	U	D	I	R	U	D
contain			X	X								
credit_debit_card												
customizable_product												
have												
manager												
payment_shoppin_cart							X	X				
picture											X	X
product												
text_table											X	X
user_address												
user_phone_number												
user_table												

	T15: Insert user				T16: Insert phone number				T17: Insert user address			
	I	R	U	D	I	R	U	D	I	R	U	D
contain												
credit_debit_card												
customizable_product												
have												
manager												
payment_shoppin_cart												
picture												
product												
text_table												
user_address									X			
user_phone_number					X							
user_table	X											
	T18: Insert Product				T19: Insert Manager				T20: Insert Text			
	I	R	U	D	I	R	U	D	I	R	U	D
contain												
credit_debit_card												
customizable_product												
have												
manager					X							
payment_shoppin_cart												
picture												
product	X											
text_table									X			
user_address												
user_phone_number												
user_table												
	T21: Insert Picture				T22: Insert Credit Debit/Card				T23: Insert Shopping Cart			
	I	R	U	D	I	R	U	D	I	R	U	D
contain												
credit_debit_card					X							
customizable_product												
have												
manager												
payment_shoppin_cart									X			
picture	X											
product												
text_table												
user_address												
user_phone_number												
user_table												
	T24: Listing the customizable product											
	I	R	U	D								
contain												
credit_debit_card												
customizable_product		X										
have												
manager												
payment_shoppin_cart												
picture												
product		X										
text_table												
user_address												
user_phone_number												
user_table												

Index Discussion

We discussed workload as shown in the tables above. Then we decided following:

- We observed that there are three most used tables, which are payment_shopping_cart, product, and user_table. That is why we decided to create indexes according to these tables.
- Since our DBMS only supports four types of indexing, we considered the workload table above to decide which indexing strategy to use in our DBMS. We used binary tree index for payment_shopping_cart and product tables since they have equality and greater/smaller conditions in queries. For user_table, we use hash indexing since we only used equality conditions at user_table queries.

Physical Design

	user_name [PK] character varying (50)	password character varying (50)	first_name character varying (50)	last_name character varying (50)	e_mail character varying (30)
1	sergu	123456789	Sergüney	Gümüş	sergu_g@outlook.com
2	iremc	987654321	İrem	Çiloğlu	irem.ciloglu@metu.edu.tr

	phone_no [PK] numeric	user_name [PK] character varying (50)
1	5351234567	sergu
2	5079876543	iremc

	address [PK] character varying (200)	user_name character varying (50)
1	METU NCC 3rd Dormitory	sergu
2	METU NCC 1st Dormitory	iremc

	t_customization_id [PK] integer	t_customization_cost numeric	t_description character varying (250)	font character varying (30)	t_size numeric	t_color character varying (30)	entered_text character varying (250)	t_approve integer
1	1	30	Front left of sweatshirt	Times New Roman	140.0	white	YOLO	1
2	1	0.00	[null]	Times New Roman	150.0	Black	[null]	[null]












	shopping_cart_id [PK] integer	product_id integer
--	----------------------------------	-----------------------

















	t_customization_id [PK] integer	t_customization_cost numeric	t_description character varying (250)	font character varying (30)	t_size numeric	t_color character varying (30)	entered_text character varying (250)	t_approve integer
1	1	30	Front left of sweatshirt	Times New Roman	140.0	white	YOLO	1
2	1	0.00	[null]	Times New Roman	150.0	Black	[null]	[null]















	p_customization_id integer	image_data_path character varying	p_description character varying (250)	p_customization_cost numeric	p_approve integer	print_size integer	t_customization_id integer	t_customization_cost numeric	t_description character varying (250)	font character varying (30)
1	1	[null]	[null]	0.00	[null]	[null]	1	0.00	[null]	Times New Roman







t_size numeric	t_color character varying (30)	entered_text character varying (250)	t_approve integer	product_id [PK] integer	product_description character varying (250)	price numeric	quantity integer	product_name character varying (30)	created_at date	updated_at date	product_type character varying (30)
150.0	Black	[null]	[null]	1	mottled crop sweatshirt	70	20	sweatshirt	2022-05-20	2022-05-20	clothes



















p_color character varying (30)	p_size character varying (30)	total_price numeric
blue	S	70.00





 p_customization_id [PK] integer 	 image_data_path character varying 	 p_description character varying (250) 	 p_customization_cost numeric 	 p_approve integer 	 print_size integer 	
1	2	https://drive.google.com/file/d/1PYKGfJRV6O819QCEkDuGXT8qZ85Of8Q/view?usp=sharing	middle at the front of product	30	1	15
2	1	[null]	[null]	0.00	[null]	[null]

 shopping_cart_id [PK] integer 	 payment_amount numeric 	 payment_address character varying (200) 	 payment_description character varying (250) 	 payment_date date 	 number_of_products integer 	 total_cost numeric 	 have character varying (50) 	
1	1	130	METU NCC 3rd Dormitory	successfully completed	2022-05-20	1	130	sergu

 manager_id [PK] integer 	 phone_no numeric 	 first_name character varying (50) 	 last_name character varying (50) 	 e_mail character varying (30) 	 password character varying (50) 	 address character varying (200) 	
1	1	5071923847	Umut	Baybece	umut.baybece@metu.edu.tr	789654123	METU NCC 2nd Dormitory

 p_customization_id [PK] integer 	 t_customization_id [PK] integer 	 product_id [PK] integer 
---	---	---

 p_customization_id integer 	 image_data_path character varying 	 p_description character varying (250) 	 p_customization_cost numeric 	 p_approve integer 	 print_size integer 	 t_customization_id integer 	 t_customization_cost numeric 	 t_description character varying (250) 
--	---	---	--	---	--	--	--	---

 credit_debit_card numeric 	 user_name character varying (50) 	
1	1234567812345678	sergu

Sample Queries

SELECT product_name FROM Product

WHERE quantity > 0;

SELECT product_name, price FROM Product

WHERE product_type = 'Clothes';

SELECT h.p_customization_id, h.product_id FROM Have h

WHERE EXISTS(SELECT t.t_approve, p.p_approve FROM Text_Table t, Picture p

WHERE h.p_customization_id = t.t_customization_id

OR h.p_customization_id = p.p_customization_id);

SELECT p.product_id, p.product_name, p.price, p.p_color, p.p_size, p.total_price FROM Contain c, Product p
WHERE c.shopping_cart_id='12345';

SELECT payment_amount,payment_address,payment_description,payment_date,number_of_products FROM
Payment_Shopping_Cart s WHERE s.have='iremc';

```
SELECT p_color,p_size,product_name,product_description,total_price,product_type
FROM Product p
WHERE product_type = '?'
ORDER BY price DESC;
```

```
SELECT a.address
FROM User_Address a,User_Table u
WHERE a.user_name = u.user_name;
```

--User_Table--

```
INSERT INTO User_Table (user_name, password, first_name, last_name, e_mail)
VALUES ('sergu', '123456789','Sergüney', 'Gümüş', 'sergu_g@outlook.com');
```

```
INSERT INTO User_Table (user_name, password, first_name, last_name, e_mail)
VALUES ('iremc', '987654321','irem','Çiloğlu','irem.ciloglu@metu.edu.tr');
```

--user_phone_number--

```
INSERT INTO User_Phone_Number (phone_no, user_name) VALUES (05351234567,'sergu');
INSERT INTO User_Phone_Number (phone_no, user_name) VALUES (05079876543,'iremc');
```

--user_address--

```
INSERT INTO User_Address (address, user_name) VALUES ('METU NCC 3rd Dormitory','sergu');
INSERT INTO User_Address (address, user_name) VALUES ('METU NCC 1st Dormitory','iremc');
```

```
DELETE FROM user_address WHERE user_name ='sergu';--deleting a user
DELETE FROM credit_debit_card WHERE user_name ='sergu';--deleting a user
DELETE FROM user_table WHERE user_name ='sergu';--deleting a user
```

UPDATE Product SET product_description='crop tshirt' WHERE product_id=1 ;--updating product_description

UPDATE Product SET price=20 WHERE product_id=1;--updating price

UPDATE Product SET quantity=5 WHERE product_id=1;--updating quantity

UPDATE Product SET p_color='red' WHERE product_id=1;--updating color

Interface and Related Queries

1)

The screenshot shows a web interface for a system named 'YMS'. At the top, there is a red header bar with 'YMS' on the left and a 'Login' link on the right. Below the header is a 'Registration form' box. The form has a light gray background and contains the following fields: 'First name:', 'Surname:', 'User name:', 'Password:', 'Re-enter password:', 'E-mail:', 'Phone number:', and 'Address:'. Each field has a corresponding text input box. At the bottom of the form is a red button labeled 'Register'.

```
cursor.execute("""INSERT INTO User_Table (user_name, password, first_name,
last_name, e_mail)
VALUES(%s, crypt(%s, gen_salt('bf')), %s, %s, %s)""",
[user_object.get_username(), user_object.get_password(),
user_object.get_first_name(),
user_object.get_last_name(), user_object.get_email()])
cursor.execute("""INSERT INTO User_Phone_Number (phone_no, user_name) VALUES(%s,
%s)""",
[user_object.get_phone_no(), user_object.get_username()])
cursor.execute("""INSERT INTO User_Address (address, user_name) VALUES (%s,
%s)""",
[user_object.get_address(), user_object.get_username()])
```

We take new user's details to register user into the YMS database to allow the him/her log in with INSERT query.

2)

YMS

SignUp

LogIn

User name:

Password:

Login

```
cursor.execute("""SELECT * FROM User_Table WHERE user_name = %s
AND password = crypt(%s, password)""", [self.__username, self.__password])
```

We let the user to log in to the YMS website with SELECT query which takes username and password which is encrypted.

3)

YMS


New Releases

Categories


Manager

Login


SignUp




t-shirt
S
70 TL




sweatshirt
M
150 TL




hoodie
L
150 TL




laptop backpack
None
120 TL




tote bag
None
50 TL




Cup
None
28 TL




thermos
None
40 TL



Flashlight
None
20 TL



Flashlight
None
20 TL



pen
None
10 TL

```
cursor.execute("""SELECT product_id, product_description, price, quantity,
product_name,
created_at, updated_at, product_type, p_color, p_size, product_image_file,
total_price FROM Product
WHERE quantity > 0""")
```

By using SELECT query, we show to the user all available products with their features in the database.

4)



```
cursor.execute("""SELECT product_id, product_description, price, quantity,
product_name,
created_at, updated_at, product_type, p_color, p_size, product_image_file,
total_price FROM Product
WHERE quantity > 0 AND product type = 'Clothes'""")
```

With SELECT query, we show the existing products and their features which belong to the selected category which is Clothes.

5)



```
cursor.execute("""SELECT product_id, product_description, price, quantity,
product_name,
created_at, updated_at, product_type, p_color, p_size, product_image_file,
total_price FROM Product
WHERE quantity > 0 AND product_type='Glassware'""")
```

With SELECT query, we show the existing products and their features which belong to the selected category which is Glassware.

6)



```
cursor.execute("""SELECT product_id, product_description, price, quantity,
product_name,
created_at, updated_at, product_type, p_color, p_size, product_image_file,
total_price FROM Product
WHERE quantity > 0 AND product_type='Bag'""")
```

With SELECT query, we show the existing products and their features which belong to the selected category which is Bag.

7)



```
cursor.execute("""SELECT product_id, product_description, price, quantity,
product_name,
created_at, updated_at, product_type, p_color, p_size, product_image_file,
total_price FROM Product
WHERE quantity > 0 AND product_type='Accessories'""")
```

With SELECT query, we show the existing products and their features which belong to the selected category which is Accessories.

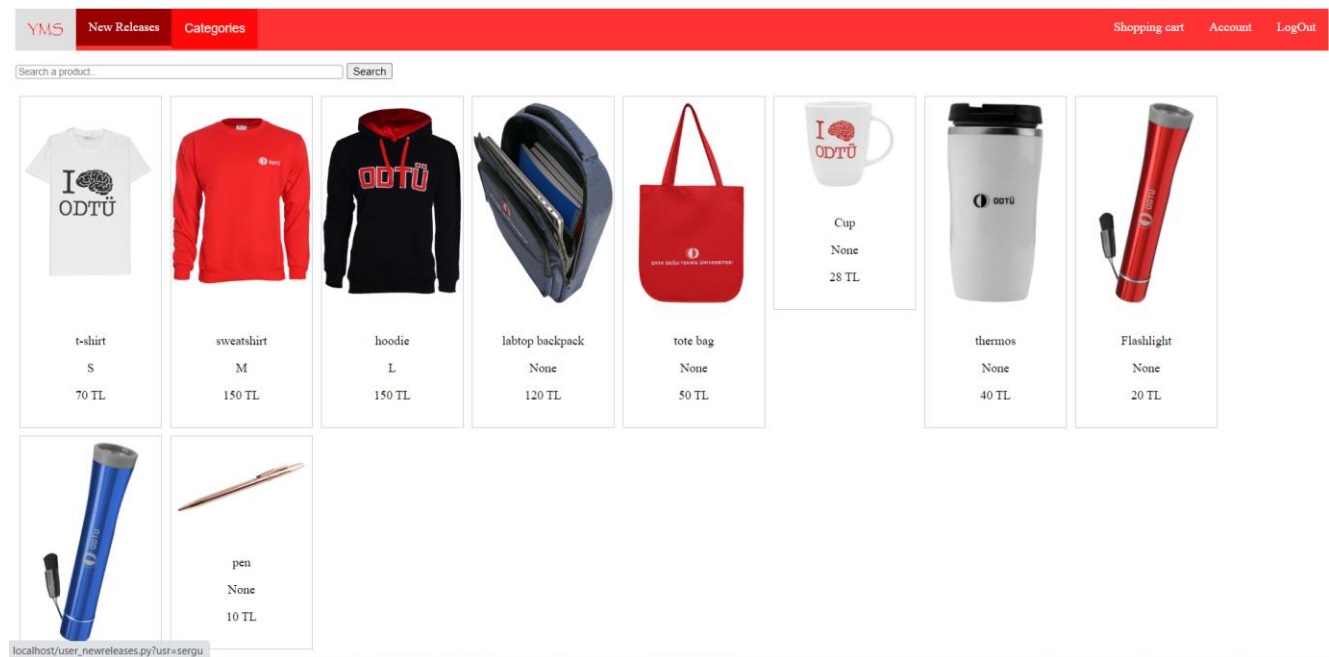
8)



```
cursor.execute("""SELECT product_id, product_description, price, quantity,
product_name,
created_at, updated_at, product_type, p_color, p_size, product_image_file,
total_price FROM Product
WHERE quantity > 0 AND product_name LIKE '{}' or product_name LIKE '{}' or
product_name LIKE '{}'""")
.format(keyword, keyword, keyword)
```

With SELECT query, we let user to search product which is pen at this figure above by their product names.

9)



```
cursor.execute("""SELECT product_id, product_description, price, quantity,
product_name,
created_at, updated_at, product_type, p_color, p_size, product_image_file,
total_price FROM Product
WHERE quantity > 0
ORDER BY created_at DESC;""")
```

With SELECT query, we display the products which are newly released to the user.

10)



```
cursor.execute("""SELECT product_id, shopping_cart_id FROM Payment_Shopping_Cart
WHERE have = %s""", [usr])
```

We let user to see the products that s/he added to the shopping cart and to buy these products.

11)



```
cursor.execute("""SELECT * FROM User Table WHERE user_name = %s""", [usr])
```

We let users to see their account details in the account page by using SELECT query.

12)



```
cursor.execute("""DELETE FROM Payment_Shopping_Cart WHERE have = %s""", [usr])
cursor.execute("""DELETE FROM User_Phone_Number WHERE user_name = %s""", [usr])
cursor.execute("""DELETE FROM User_Address WHERE user_name = %s""", [usr])
cursor.execute("""DELETE FROM Credit_Debit_Card WHERE user_name = %s""", [usr])
cursor.execute("""DELETE FROM User_Table WHERE user_name = %s""", [usr])
```

We allow the users to deactivate their accounts in the system and we delete the account from the database with DELETE queries.

13)

YMS

Management System

E-mail:

Password:

Login

```
cursor.execute("""SELECT * FROM Manager WHERE e_mail = %s
AND password = crypt(%s, password)""", [self.__e_mail, self.__password])
```

The managers can log in the system with Management System log in page by SELECT query.

14)

YMS PRODUCTS CUSTOMIZATION APPROVAL LogOut

Product Name
Enter Product Name

Product Description
Product Description

Product Quantity
Product Quantity

Product Type
Product Type

Product Color
Product Color

Product Size
Product Size

Product Price
Product Price

Product Link
Product Image Link

SUBMIT

Product ID
Enter Product ID

Product Quantity
Enter Product Quantity

UPDATE

Product ID
Enter Product ID

DELETE

```
cursor.execute("""INSERT INTO Product (product_description, price, quantity,
product_name,
product_type, p_color, p_size, product_image_file)
VALUES(%s, %s, %s, %s, %s, %s, %s, %s)""", (p.get_description(),
float(p.get_price()),
float(p.get_quantity()), p.get_name(),
p.get_type(),
p.get_color(), p.get_size(),
p.get_img(),))
```

The webpage allows the manager to add new products to the database with INSERT query in manage product page.

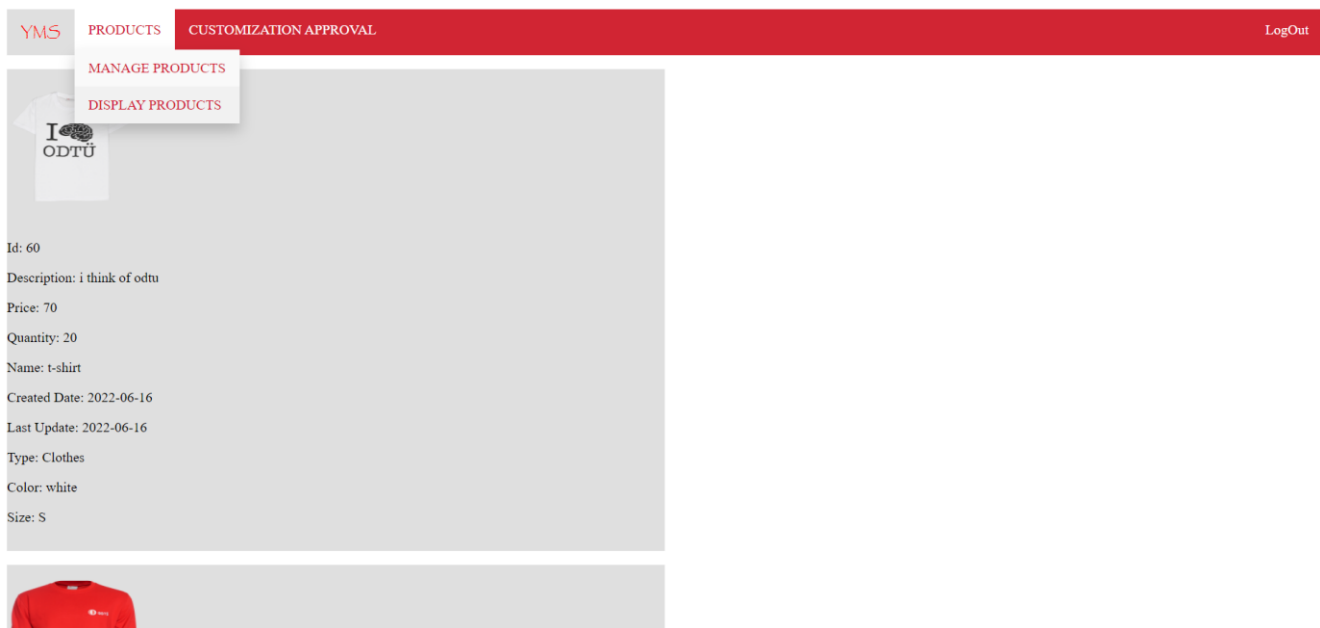
```
cursor.execute("""DELETE FROM Product WHERE product_id=%s""", (id,))
```

The webpage allows the manager to delete products from the database with DELETE query.

```
cursor.execute("""UPDATE Product SET quantity=%s WHERE product_id=%s""",
(variable, id,))
```

The webpage allows the manager to update products to the database with UPDATE query.

15)



```
cursor.execute("""SELECT product_id, product_description, price, quantity,
product_name,
created_at, updated_at, product_type, p_color, p_size, product_image_file,
total_price FROM Product""")
```

We allow the manager to see the all products in the database system in display products page.