

Electronic Vendor Database

Group 21

Hyunsuk Bang

George Tapia

Marcos Ferreira

Contents

Introduction	3
ER – Diagram	4
Getting the application to Run.....	5
Signing Up	6
Adding a Credit Card:	7
Adding a Address:	8
Home Page	9
Selected Product	10
Shopping Cart.....	10
Checkout	11
Purchase History	12
Store and Warehouse Administrators	12
Member Contributions	15

Introduction

The electronic vendor we built is similar to 'Best Buy'. It has the standard features such as creating an account, querying different products, modifying/viewing account information (including keeping track of credit card and address information), purchasing history, shopping cart and so on... Since our database simulates multiple stores connected to a common warehouse, we also have store and warehouse admins to manage the inventories.

All the data is stored in a mySQL database. Our application then fetches the appropriate information from the database and presents it to the customer (along with other interactions) with a easy to use user interface. To see more detail of the database, look at the ER diagram just below.

The diagram illustrates a database schema for a supply chain management system. It consists of 20 tables, each with its fields and data types listed. The tables are interconnected by lines representing relationships, with many-to-one relationships indicated by crow's foot notation (a '1' at the end of the line and a crow's foot symbol at the other end).

Tables and their fields:

- memberAddress**: m_id (BIGINT), address1 (varchar(50)), address2 (varchar(50)), state (varchar(2)), zipcode (varchar(15)).
- member**: m_id (BIGINT), name (varchar(20)), phone (varchar(20)), email (varchar(50)), type (member_type_enum), user_status (int), reg_date (DATE), billing_date (DATE).
- warehouse**: w_id (BIGINT), address1 (varchar(20)), state (VARCHAR(2)), zipcode (varchar(15)).
- warehouseAdmin**: wh_a_id (BIGINT), w_id (BIGINT), name (varchar(20)), phone (varchar(20)), email (varchar(50)).
- warehouseInventory**: id (BIGINT), w_id (BIGINT), p_id (BIGINT), quantity (int), threshold (int).
- store**: s_id (BIGINT), address (VARCHAR(20)), state (varchar(2)), zipcode (varchar(15)).
- storeAdmin**: store_a_id (BIGINT), s_id (BIGINT), name (varchar(20)), phone (varchar(20)), email (varchar(50)).
- storeInventory**: id (BIGINT), s_id (BIGINT), p_id (BIGINT), quantity (int), threshold (int).
- storeOrder**: s_id (BIGINT), w_id (BIGINT), p_id (BIGINT), quantity (int), reorderDate (date).
- warehouseOrder**: w_id (BIGINT), manufacturer_id (BIGINT), p_id (BIGINT), quantity (int), reorderDate (date).
- storeAdmin**: store_a_id (BIGINT), s_id (BIGINT), name (varchar(20)), phone (varchar(20)), email (varchar(50)).
- restockWarehouse**: w_id (BIGINT), manufacturer_id (BIGINT), p_id (BIGINT), quantity (BIGINT), restock_date (date).
- restockStore**: s_id (BIGINT), w_id (BIGINT), p_id (BIGINT), quantity (BIGINT), restock_date (date).
- admin**: a_id (BIGINT), s_id (BIGINT), name (varchar(20)), phone (varchar(20)), email (varchar(50)).
- whStore**: w_id (BIGINT), s_id (BIGINT).
- manufacturer**: manufacturer_id (BIGINT), manufacturer_name (varchar(100)), email (varchar(30)), phone_num (varchar(20)).
- product**: p_id (BIGINT), category (BIGINT), p_name (varchar(50)), wholesale_price (float), instore_price (float), manufacturer_id (BIGINT).
- category**: category (BIGINT).
- onlineOrder**: id (BIGINT), order_id (BIGINT), order_date (date), p_id (BIGINT), quantity (int), customer_type (int), m_id (BIGINT), email (varchar(30)), card_info (varchar(20)), address1 (varchar(50)), address2 (varchar(50)), state (varchar(2)), zip_code (varchar(15)), phone_num (BIGINT), recipient_name (varchar(30)), recipient_phone (BIGINT), sc_id (BIGINT), tracking_num (varchar(50)).
- shippingCompany**: sc_id (BIGINT), sc_name (varchar(50)).
- orderList**: order_id (BIGINT), order_date (date).

Relationships:

- memberAddress (1) to member (many).
- member (1) to warehouse (many).
- member (1) to store (many).
- warehouse (1) to warehouseAdmin (many).
- warehouse (1) to warehouseInventory (many).
- store (1) to storeAdmin (many).
- store (1) to storeInventory (many).
- store (1) to storeOrder (many).
- warehouse (1) to warehouseOrder (many).
- store (1) to restockWarehouse (many).
- store (1) to restockStore (many).
- admin (1) to store (many).
- whStore (1) to warehouse (many).
- whStore (1) to store (many).
- manufacturer (1) to product (many).
- product (1) to category (many).
- product (1) to warehouseOrder (many).
- product (1) to storeOrder (many).
- product (1) to restockWarehouse (many).
- product (1) to restockStore (many).
- onlineOrder (1) to member (many).
- onlineOrder (1) to warehouse (many).
- onlineOrder (1) to store (many).
- onlineOrder (1) to shippingCompany (many).
- onlineOrder (1) to orderList (many).

Getting the application to Run

1. Download the linked source files
2. Install Python (<https://www.python.org/downloads/>)
3. Install 'Django' (<https://docs.djangoproject.com/en/4.0/topics/install/>)
4. Create a mySQL database and run the provided SQL script on it
 - a. Make sure to run the create table statements with trigger statement **and then** the insert statements
5. Within the source file, navigate into '/CS425_FINAL_PROJECT/mysite' and open 'settings.py'
 - a. Edit the username, password and DB name to match your mysql username, password and DB name

```
82 DATABASES = {  
83     'default': {  
84         'ENGINE': 'django.db.backends.mysql',  
85         'NAME': 'your_database_name',  
86         'USER': 'your_username',  
87         'PASSWORD': 'your_password',  
88         'HOST': 'localhost',  
89     }  
90 }
```

6. Navigate back to '/CS425_FINAL_PROJECT' and run on the command prompt:
 - a. python manage.py makemigrations
 - b. python manage.py migrate
 - c. python manage.py createsuperuser
 - i. this is important so that you can access the DB admin page which in the URL is at './admin'
 - d. python manage.py runserver
 - i. copy and paste the http address to access the website

Note! There maybe one or two libraries that are not installed on your machine, please read which library the error message says is missing and install it using 'pip' installer (<https://pip.pypa.io/en/stable/installation/>)

Signing Up

Here I will be showing the step-by-step process that a customer would do to shop on our website, starting with creating an account. However, creating an account is not necessary, as the user can also shop as a guest (which will be demonstrated later).

sign_up

ID:

Password:

retype Password:

email:

Name:

phone:

type

[go back](#)

sign_up

ID:

Password:

retype Password:

email:

Name:

phone:

type

[go back](#)

There are two types of users; normal users and subscript users. User's who subscribe to our website will have their accounts build monthly while normal users will be charged immediately for any purchases made.

Log IN

! username or password is incorrect. or user id does not exist

ID:

Password:

[go back](#)

After creating an account, logging in with the wrong credentials will greet the user with this error message.

After successfully logging in, the user will be able to view and modify his/her account details as demonstrated below.

ID	email	phone	registration date
122	example122@gmail.com	123456789	May 4, 2022

new email:

new phone: number

type

[go back](#)

ID	email	phone	registration date
122	another122@gmail.com	111222333	May 4, 2022

new email:

new phone: number

type

[go back](#)

The user can also add different credit cards and addresses

[Adding a Credit Card:](#)

add new card

card number:

card holder name:

exp_month:

exp_year:

[go back](#)

After adding two cards, the user will be able to view their balances and delete them if user want's to protect their privacy and not save their credit card information on our website.

	card number	name on card	card expiration	Balance	
1	2000312288719332	customer122	8 / 27	614576.2 USD	<button>delete</button>
2	9822303076652211	customercousin122	7 / 27	345163.2 USD	<button>delete</button>

[new card](#)
[go back](#)

Adding a Address:

add new address

street address:

Box, Apt Num:

State:

zipcope:

[go back](#)

After adding three addresses, the user can view or delete them.

	address	state	zipcode	
1	JJohns Street 2	IL	60616	<button>delete</button>
2	Jupiter street 5	NY	101010	<button>delete</button>
3	Marsville street 2	NY	101010	<button>delete</button>

[new address](#) [go back](#)

After fully setting up an account, now we will explore the *home page*:

Home Page

Here is the home page. Functionalities include links to, editing account info, seeing purchase history, viewing/modifying cards, viewing/modifying addresses, and the shopping cart.

Hello, customer122
log out
edit my info
my purchase history
my cards
my address
cart

	Product Name	Category category	Manufacturer manufacturer	<div>Apply</div>	Price
1	dell book	computer	Dell		1300.0 USD
2	dell book pro	computer	Dell		1800.0 USD
3	Lenovo Carbon	computer	Lenovo		1900.0 USD
4	Lenovo Think pad	computer	Lenovo		2600.0 USD
5	Logitech keyboard	accessory	Logitech		250.0 USD
6	Logitech keyboard pro	accessory	Logitech		450.0 USD
7	Logitech mouse	accessory	Logitech		100.0 USD
8	Razer blade 13	computer	Razer		1700.0 USD
9	Razer blade 15	computer	Razer		1900.0 USD
10	google pixel 6	phone	Google		1400.0 USD
11	galaxy s 22	phone	Samsung		1300.0 USD
12	google pixel 4a	phone	Google		900.0 USD
13	xperia xi	phone	Sony		1400.0 USD
14	sony xdr cam	camera	Sony		1700.0 USD
15	sony dslr	camera	Sony		3200.0 USD
16	canon pro	camera	Canon		3300.0 USD
17	canon beginner	camera	Canon		1400.0 USD
18	galaxy z flip	phone	Samsung		1400.0 USD
19	galaxy tab	tablet	Samsung		1450.0 USD
20	macbook air	computer	Apple		1000.0 USD
21	macbook pro	computer	Apple		1700.0 USD
22	macbook studio	computer	Apple		2700.0 USD
23	iphone14	phone	Apple		1400.0 USD

We also see that we can query the products according to their category and manufacture. For example, if we just want to see computer from all manufactures:

Product Name	Category computer ▾	Manufacturer manufacturer ▾	Apply	Price	
1 dell book	computer	Dell		1300.0 USD	
2 dell book pro	computer	Dell		1800.0 USD	
3 Lenovo Carbon	computer	Lenovo		1900.0 USD	
4 Lenovo Think pad	computer	Lenovo		2600.0 USD	
5 Razer blade 13	computer	Razer		1700.0 USD	
6 Razer blade 15	computer	Razer		1900.0 USD	
7 macbook air	computer	Apple		1000.0 USD	
8 macbook pro	computer	Apple		1700.0 USD	
9 macbook studio	computer	Apple		2700.0 USD	

Or instead, say we only want Apple computers:

Product Name		Category	Manufacturer	Apply	Price
		computer ▾	Apple ▾		
1	macbook air	computer	Apple		1000.0 USD
2	macbook pro	computer	Apple		1700.0 USD
3	macbook studio	computer	Apple		2700.0 USD

Selected Product

After selecting a product, the user can then select the quantity that they want to add to their cart.

Name	manufacturer	category	Price
macbook pro	Apple	computer	1700.0

Low in Stock!
18 left

add this to cart

quantity:

3

add to cart

[go back](#)

Shopping Cart

Once done shopping, the user can then view their shopping cart and either adjust the quantities or remove undesired products.

Product	unit price	qunatity	total	update quantity		
1 Logitech keyboard pro	450.0 USD	1	450.0 USD	+	-	delete
2 canon pro	3300.0 USD	1	3300.0 USD	+	-	delete
3 macbook pro	1700.0 USD	3	5100.0 USD	+	-	delete

[go back](#)

proceed to check out

Product	unit price	qunatity	total	update quantity		
1 Logitech keyboard pro	450.0 USD	2	900.0 USD	+	-	delete
2 canon pro	3300.0 USD	1	3300.0 USD	+	-	delete
3 macbook pro	1700.0 USD	2	3400.0 USD	+	-	delete

[go back](#)

proceed to check out

Checkout

For a logged in user, the user will choose the card and address out of the ones they saved on their account (of course they can always delete the cards and addresses later if they want their data privacy respected).

Summary:

	Product	unit price	qunatity	total
1	Logitech keyboard pro	450.0 USD	2	900.0 USD
2	canon pro	3300.0 USD	1	3300.0 USD
3	macbook pro	1700.0 USD	2	3400.0 USD

total: 7600.0 USD

Card

Address

recipient

recipient name: recipient phone:

[place order](#)

[Main page](#)

For a guest (user that's not logged in), the user will need to fill in the card and address details (but these details will not be permanently stored on the database).

Summary:

	Product	unit price	qunatity	total
1	Logitech keyboard pro	450.0 USD	1	450.0 USD
2	canon pro	3300.0 USD	1	3300.0 USD
3	macbook pro	1700.0 USD	1	1700.0 USD

total: 5450.0 USD

card

card number:

card holder name:

exp_month:

exp_year:

Address

street address:

Box, Apt Num:

State:

zipcode:

Phone

phone:

recipient

recipient name: recipient phone:

[place order](#)

After placing order, the user will be given their tracking number (for shipping).

Thank you for your purchase!

your tracking number is 2022-05-04 20:30:51.458220+00:001

[Main page](#)

Purchase History

For users with accounts, they can then view their own purchase history:

	date	product	quantity	card	address	phone
1	May 4, 2022	Logitech keyboard pro	2	2000312288719332	Jupiter street	111222333
2	May 4, 2022	canon pro	1	2000312288719332	Jupiter street	111222333
3	May 4, 2022	macbook pro	2	2000312288719332	Jupiter street	111222333

[Main page](#)

Store and Warehouse Administrators

To manage the store and warehouse, we will need store and warehouse admins. Setting up these accounts is the responsibility of the database administrator. The DB admin will first register store and warehouse accounts with specific IDs.

The screenshot shows the Django administration interface. On the left, a sidebar menu includes 'AUTHENTICATION AND AUTHORIZATION' with 'Groups' and 'Users' links, and 'USER' with 'Store admins' and 'Warehouse admins' links. The main content area displays the 'Add store admin' form with fields for 'Store a id' (storeAdmin_s1), 'S' (Store object (s,1)), 'Name' (storeAdmin_s1), 'Phone' (123456779), and 'Email' (storeadmin_s1@gmail.com). At the bottom are 'Save and add another', 'Save and continue editing', and 'SAVE' buttons. A 'Django administration' login modal is overlaid on the right, with fields for 'Username' (admin) and 'Password' (masked with dots), and a 'Log in' button.

sign_up

The respective store/warehouse admins will then *sign up* using the same ID (e.g in this example for a store admin, the ID is 'storeAdmin_s1')

The screenshot shows a sign-up form with the following fields: 'ID:' (storeAdmin_s1), 'Password:' (masked with dots), 'retype Password:' (masked with dots), 'email:' (storeadmin_s1@gmail.com), 'Name:' (storeadmin_s1), 'phone:' (123456779), and a 'type' dropdown menu (Normal). At the bottom are a 'Sign Up' button and a 'go back' link.

Upon logging in with a store admin account, the store admin will be show a page that will enable them to request inventory from the warehouse and a check will be done to ensure that don't request more than is available.

	Product	Quantity	Price
1	dell book	1	1300.0 USD
2	dell book pro	19	1800.0 USD
3	Lenovo Carbon	6	1900.0 USD
4	Logitech keyboard	2	250.0 USD
5	Logitech keyboard pro	9	450.0 USD
6	Logitech mouse	12	100.0 USD
7	galaxy s 22	7	1300.0 USD
8	galaxy z flip	2	1400.0 USD
9	macbook air	5	1000.0 USD
10	macbook pro	7	1700.0 USD
11	macbook studio	3	2700.0 USD

✓ add success!

galaxy s 22-----18 left

3

Request

logout

	wareHouse Inventory	quantity
1	galaxy s 22	18
2	galaxy z flip	20
3	galaxy tab	20
4	macbook air	20
5	macbook pro	18
6	macbook studio	20
7	lphone14	20
8	lpad	20
9	dell book	20
10	dell book pro	10
11	Lenovo Carbon	20
12	Lenovo Think pad	20
13	Logitech keyboard	20
14	Logitech keyboard pro	18
15	Logitech mouse	20
16	Razer blade 13	20
17	Razer blade 15	20
18	google pixel 6	40
19	google pixel 4a	20
20	xperia xi	20
21	sony xdr cam	20
22	sony dslr	20
23	canon pro	19
24	canon beginner	20

⚠ can't request more than what warehouse have

galaxy s 22-----20 left

100

Request

logout

This is similar to the warehouse admin, except the warehouse admin order inventory from the manufacturers. Since we don't know how much inventory the manufacturers have, we assumed there to be no cap on the orders.


	Product Name	quantity	manufacturer
1	galaxy s 22	25	Samsung
2	galaxy z flip	20	Samsung
3	galaxy tab	20	Samsung
4	macbook air	20	Apple
5	macbook pro	18	Apple
6	macbook studio	20	Apple
7	Iphone14	20	Apple
8	Ipad	20	Apple
9	dell book	20	Dell
10	dell book pro	10	Dell
11	Lenovo Carbon	20	Lenovo
12	Lenovo Think pad	20	Lenovo
13	Logitech keyboard	20	Logitech
14	Logitech keyboard pro	18	Logitech
15	Logitech mouse	20	Logitech
16	Razer blade 13	20	Razer
17	Razer blade 15	20	Razer
18	google pixel 6	40	Google
19	google pixel 4a	20	Google
20	xperia xi	20	Sony
21	sony xdr cam	20	Sony
22	sony dslr	20	Sony
23	canon pro	19	Canon
24	canon beginner	20	Canon

galaxy s 22-----25 left

20

Request

logout

 success!

We can also see the record of the requested orders from both the store and warehouse admins on the database with the date stamps:

s_id	w_id	p_id	quantity	restock_date
s_1 →	w_1 →	p_16 →	4	2022-05-03 ↕
s_1 →	w_1 →	p_16 →	4	2022-05-03 ↕
s_1 →	w_1 →	p_14 →	3	2022-05-03 ↕
s_1 →	w_1 →	p_14 →	5	2022-05-03 ↕
s_1 →	w_1 →	p_18 →	10	2022-05-03 ↕
s_1 →	w_1 →	p_9 →	3	2022-05-03 ↕
s_1 →	w_1 →	p_9 →	4	2022-05-03 ↕
s_1 →	w_1 →	p_16 →	2	2022-05-03 ↕
s_1 →	w_1 →	p_16 →	2	2022-05-03 ↕
s_1 →	w_1 →	p_5 →	3	2022-05-03 ↕
s_1 →	w_1 →	p_21 →	10	2022-05-03 ↕
s_1 →	w_1 →	p_19 →	3	2022-05-03 ↕
s_1 →	w_1 →	p_7 →	10	2022-05-03 ↕
s_1 →	w_1 →	p_6 →	10	2022-05-03 ↕
s_1 →	w_1 →	p_3 →	5	2022-05-03 ↕
s_1 →	w_1 →	p_3 →	100	2022-05-03 ↕
s_1 →	w_1 →	p_3 →	100	2022-05-04 ↕
s_1 →	w_1 →	p_3 →	100	2022-05-04 ↕
s_1 →	w_1 →	p_7 →	9	2022-05-04 ↕
s_1 →	w_1 →	p_20 →	10	2022-05-04 ↕

w_id	manufacturer_id	p_id	quantity	restock_date
w_1 →	man_01 →	p_3 →	1	2022-05-03 ↕
w_1 →	man_01 →	p_3 →	87	2022-05-03 ↕
w_1 →	man_01 →	p_3 →	98	2022-05-03 ↕
w_1 →	man_07 →	p_19 →	100000	2022-05-04 ↕

Member Contributions

Although we each have areas we individually focused more on, we all contributed to some degree to all areas of the project.

Hyunsuk Bang – UI design, Application Development, Database Development

George Tapia – ER diagram, Database Development

Marcos Ferreira – Application Development, Documentation, Application Testing