

ABDULLAH GÜL
UNIVERSITY

Database Management Systems

Project

Hamzah Haroun | 110210095

Lae Lae Win | 110510130

Yılmaz Şanlı | 110510046

CONTENT

CONTENT	2
Organization	3
Summary	3
Abstract	3
Requirement Analysis	3
Tools/IDEs	4
Diagram/Users	5
Entity-Relationship Diagram	5
Design Philosophy	6
High Level Design	6
E-R To Relational Mapping	7
E-R Diagram Cardinalities	7
A - 1:1 Relations	7
B - 1:N Relations	7
C - M:N Relations	7
Table Normalization	8
Database Schemas	8
1 - Customer Table	8
2- Address Table	9
3- Phone Number Table	9
4- Comment Table	10
5- Order Table	10
6- Cart Table	11
7- Product Table	11
8- Category Table	12
9- Admin Table	12
Table Creation Scripts	13
Constraints	16
Additional Scripts	18
MySQL Workbench Generated E-R Diagram	19
Datasets	20

Organization

The system being developed consists of several levels of organizations, the admin and the customer levels, the admin has access to the back data and organization of the application, while the customer has access to the relevant information, like their account information and the shopping items.

Summary

In short, the system we have intends to serve customers looking to buy items from an grocery shop, we will be using several tables to divide those items into categories by type and store each product relevant information in a table, we also intend to allow our customers to have their own accounts so they can store their information and make their own carts.

Abstract

In our system, customers can register to the system to buy items. Authenticated customers can see all items in the shop, search for specific items, can group the items according to filters and buy them. We intend to use Java as our main programming language and we will be using it on a web platform. The application development will consist of 2 main steps. The first one is creating a REST API by using Java's framework which is called Spring Boot. This API will be used to create the back-end of the program. The second step is creating a web UI by using JavaScript's framework which is called Angular.

Requirement Analysis

The main aim of this project is to develop an online market system in which consumers can buy things that they want. To create a shop database, admin, customer and products information must be stored. Customers may order products and comment on the items. An admin or shopkeeper should add or delete items to this store. In this shop database, there are different types of products. This information will be stored in a categories table. Products are groups under categories.

The online market system is one of the essential things currently. According to research, the B2C (business to consumer) and B2B (business to business) e-commerce have been growing exponentially. The growth of e-commerce creates opportunities to develop and operate small to medium size B2B or B2C e-commerce systems. The advantages of developing such systems make time saving for consumers and lower cost for companies.

Business requirements: Along with the growth of the internet, B2C online shoppings, including online markets are required for those who do not want to go out for shopping.

User requirements: Users can use websites or applications for purchasing online that is not only convenient but also time saving for them.

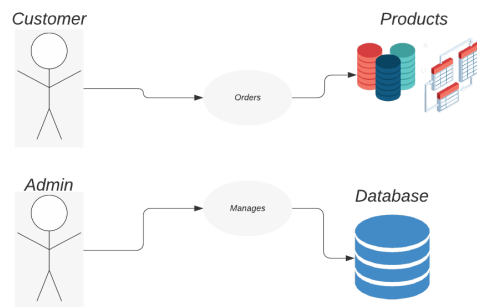
Table and requirement relations:

- A user can sign up the system so there should be a user table with which the username, password and the other information related to the user is stored.
- The website will show the products, therefore, there should be a product table which stores the information about the product.
- The product may belong to different categories and users can filter products according to categories. Therefore, there should be a category table which stores the category information.
- The registered user can add products to the carts. Therefore, there should be a cart table which holds information about product and user.
- Carts can be ordered so there should be an order table which stores the information about which cart is ordered when, cart information and date information.
- There will be an admin which adds and deletes the products. Therefore, there should be an admin table which stores information about admin information like username and password.
- Users can enter comments about products so there should be a comment table which stores information about the customer, product and comment.

Tools/IDEs

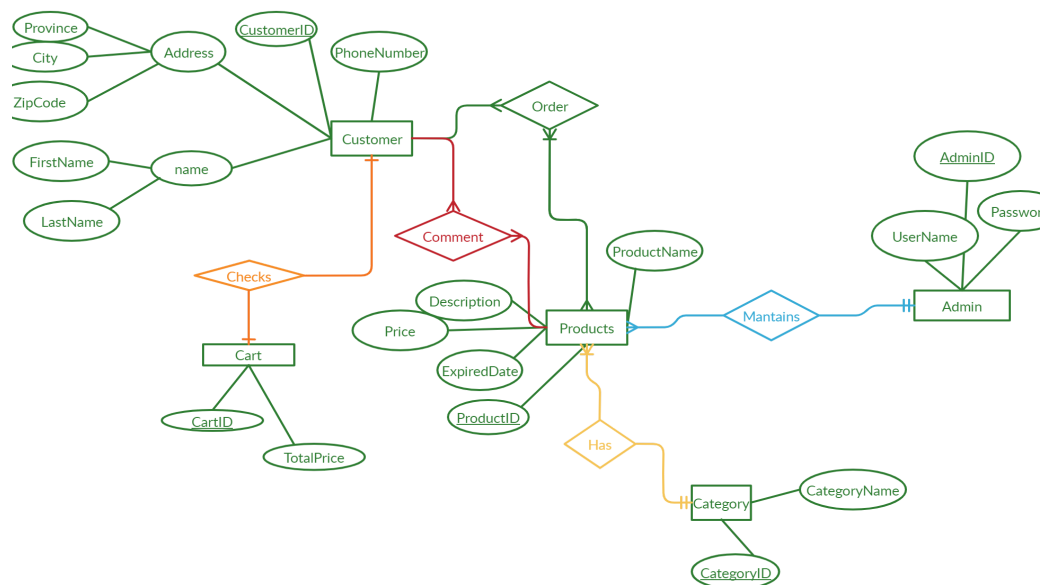
IntelliJ for API development in JAVA. Visual Studio Code for Angular development. MySQL Workbench for the database design.

Diagram/Users



This diagram shows that registered customers can order products and the admin can manage the store.

Entity-Relationship Diagram



Registered users are shown as customers. A customer can add products to cart and then order them. Therefore, there is one to many relations between customer and order because a customer can order many products. In addition, a customer can have just one cart, so there is one to one relationship between customer and cart. Customers can comment on products and the relation is one to many because a customer can add many comments on products. There is a one to many relation between category and products because many

products can be owned by one category. Finally, there is a one to many relation between admin and products because an admin can maintain many products(add or delete products).

Design Philosophy

- Category
- Product
- Customer
- Order
- Admin
- Comment
- Cart

In the category table, we will store the category information about the products.

In the product table, the information about product detail is stored. For example the name of the product, price and description.

In the customer table, the information about the customer will be stored like name, surname, phone number, address and etc. After the registration process, the details about the user will be stored here and the user can enter the system and add products to their own cart.

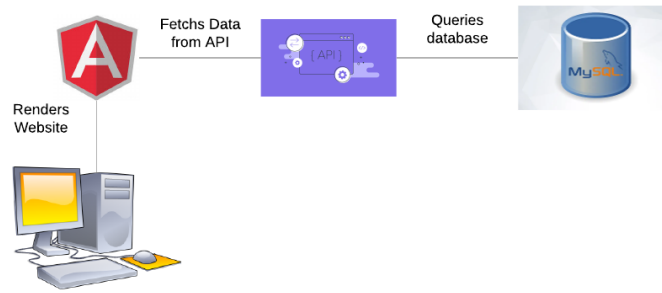
In the order table, information about user order's will be stored. After a user buys a product information is shown in this table.

In the admin table, the shopkeeper information will be stored. The admin can change the stock of the products, add new products and delete products.

In the comment table, the customer evaluations about products will be stored.

In the cart table, customers can add products in order to buy them. Information related to users and products will be stored.

High Level Design



The website will be developed by using a JavaScript framework called Angular. API gets the required information from MySQL database and this information will be fetched on Angular to render the website.

E-R To Relational Mapping

E-R Diagram Cardinalities

A - 1:1 Relations

- customer(customer_id)
- category(category_id)
- admin(admin_id)

B - 1:N Relations

- address(address_id, customer_id)
- phone_number(phone_id, customer_id)
- product(product_id, category_id)
- comment(comment_id, customer_id, product_id)
- cart(cart_id, customer_id, product_id)

C - M:N Relations

- order_table(order_id, cart_id, address_id)

Table Normalization

- In the customer table, there is no composite attribute and no functional dependency so the table is in normalized form.
- In the category table, there is no composite attribute and no functional dependency so the table is in normalized form.
- In the admin table, there is no composite attribute and no functional dependency so the table is in normalized form.
- To make the customer and address table normalized, we have created an address table because if we keep address information in the customer table then 1NF form will be violated.
- Like in the address table, if we keep phone information in the customer table the 1NF form will be violated so we have created a separate table for the phone info table.
- To make the product table normalized, we have created separate table for category. The reason is that if we keep category id and category name in the product table there will be dependency between id and name. If a tuple is entered wrongly, to change this problem we should update two entries. Therefore, the 2NF form will be violated
- To make the comment table normalized, the id of customer and product is stored here. If we store comment id in product table then we cannot manage the cardinality. Therefore, we have created separate table for comment.
- In the order table, there is a many to many relation. This cardinality cannot be implemented in SQL. Therefore, we have to break this cardinality into two one to many relation. To make the order table normalized, there should be pivot table. Then the table will be normalized.

Database Schemas

1 - Customer Table

Column	Data Type	Nullity	Entity Feature
customer_id	int	N	Primary Key
first_name	varchar	N	

last_name	varchar	N	
email_address	varchar	N	Unique

Example

customer_id	first_name	last_name	email_address
1	yilmaz	sanli	yilmaz.sanli@agu.edu.tr

2- Address Table

Column	Data Type	Nullity	Entity Feature
address_id	int	N	Primary Key
customer_id	int	N	Foreign Key
country	varchar	N	
city	varchar	N	
address	varchar	N	
zip_code	int	N	

Example

address_id	customer_id	country	city	address	zip_code
1	1	Turkey	Kayseri	Agu Sumer Campus	38010

3- Phone Number Table

Column	Data Type	Nullity	Entity Feature
phone_id	int	N	Primary Key
customer_id	int	N	Foreign Key

mobile_phone	varchar	N	
home_phone	varchar	Y	

Example

phone_id	customer_id	mobile_phone	home_phone
1	1	05555555555	NULL

4- Comment Table

Column	Data Type	Nullity	Entity Feature
comment_id	int	N	Primary Key
customer_id	int	N	Foreign Key
product_id	int	N	Foreign Key
rating	enum	N	Foreign Key
comment_date	timestamp	N	
comment_desc	varchar	Y	

Example

comment_id	customer_id	product_id	rating	comment_date	comment_desc
1	1	1	Excellent	2020-05-09 12:00:08	The laptop is very fast. I recommend this laptop!

5- Order Table

Column	Data Type	Nullity	Entity Feature
order_id	int	N	Primary Key

cart_id	int	N	Foreign Key
address_id	int	N	Foreign Key
order_date	timestamp	N	

Example

order_id	cart_id	address_id	order_date
1	1	1	2020-04-12 12:22:08

6- Cart Table

Column	Data Type	Nullity	Entity Feature
cart_id	int	N	Primary Key
customer_id	int	N	Foreign Key
product_id	int	N	Foreign Key

Example

cart_id	customer_id	product_id
1	1	1

7- Product Table

Column	Data Type	Nullity	Entity Feature
product_id	int	N	Primary Key
category_id	int	N	Foreign Key
product_name	varchar	N	Unique
product_desc	varchar	N	

product_price	int	N	
product_expire_date	date	Y	

Example

product_id	category_id	product_name	product_desc	product_price	product_expire_date
1	1	Laptop	4 core computer	2500	NULL

8- Category Table

Column	Data Type	Nullity	Entity Feature
category_id	int	N	Primary Key
category_name	varchar	N	Unique

Example

category_id	category_name
1	Electronic

9- Admin Table

Column	Data Type	Nullity	Entity Feature
admin_id	int	N	Primary Key
admin_username	varchar	N	Unique
admin_password	varchar	N	

Example

admin_id	admin_username	admin_password
1	admin	e10adc3949ba59abb e56e057f20f883e

Table Creation Scripts

Customer Table

- create table customer(
customer_id int auto_increment,
first_name varchar(50) not null,
last_name varchar(50) not null,
email_address varchar(50) not null unique,
primary key (customer_id)
);
- This is the creation of the customer table in which the customer_id is a primary key and will be auto incremented. There will be first_name, last_name and email_address as varchar. The email_address will be unique for a customer.

Address Table

- create table address(
address_id int auto_increment,
customer_id int not null,
country varchar(50) not null,
city varchar(50) not null,
address varchar(50) not null,
zip_code int not null,
primary key(address_id)
);
- This is the creation of the address table. The address will be kept as a table and it will store the customer id as a foreign key. In this table, the address_id is the primary key which will be auto incremented. The address includes country, city, address (in detail) as varchar as well as zip_code as an int.

Phone_number Table

- create table phone_number(
 phone_id int auto_increment,
 customer_id int not null,
 mobile_phone varchar(50) not null,
 home_phone varchar(50) null,
 primary key (phone_id)
);
- This is the creation of the phone_number table where the primary key is phone_id which will be auto incremented. The customer_id will be stored as foreign key. There will be mobile_phone, home_phone in varchar type as a customer may use either phone or home phone number.

Category Table

- create table category(
 category_id int auto_increment,
 category_name varchar(50) not null unique,
 primary key(category_id)
);
- This is the creation of the category table where the category_id is kept as a primary key which will be auto incremented. The category will have category_name in varchar type.

Product Table

- create table product(
 product_id int auto_increment,
 category_id int not null,
 product_name varchar(50) not null unique,
 product_desc varchar(500) not null,
 product_price int not null,
 product_expire_date date,

- ```

 primary key (product_id)
);

```
- This is the creation of the product table where the product\_id is kept as a primary key which will be auto incremented. The product will include name, description which are varchar, the price as int and the expire date as the date type.

### **Cart Table**

- create table cart(

```

 cart_id int auto_increment,
 customer_id int not null,
 product_id int not null,
 primary key (cart_id)
);

```
- The cart table includes the cart\_id as a primary key and customer\_id and product id as the foreign keys in integer type that cannot be null.

### **Comment Table**

- create table comment(

```

 comment_id int auto_increment,
 customer_id int not null,
 product_id int not null,
 rating enum('Bad', 'Below Average', 'Average', 'Good', 'Excellent') not null,
 comment_date timestamp not null,
 comment_desc varchar(500),
 primary key (comment_id)
);

```
- We have a comment table in which the comment\_id is a primary key which is auto incremented. The table includes customer\_id and product\_id as foreign keys that cannot be null. There are also rating enum consisting of different levels, the comment\_date and comment\_description.

## Admin Table

- create table admin(  
    admin\_id int auto\_increment,  
    admin\_username varchar(50) not null unique,  
    admin\_password varchar(50) not null,  
    primary key (admin\_id)  
);
- The admin table includes admin\_id as a primary key as an integer that can be auto\_incremented. The table also includes the unique admin\_username that cannot be null as well as the admin\_password.

## Order\_table Table

- create table order\_table(  
    order\_id int auto\_increment,  
    cart\_id int not null,  
    address\_id int not null,  
    order\_date timestamp not null,  
    primary key (order\_id)  
);
- This is the creation of the order\_table in which the order\_id is the primary key and the cart\_id together with address\_id are kept as foreign keys that cannot be null. The order contains an order\_date.

## Constraints

- alter table address  
    add constraint FK\_CustomerAddress  
    foreign key (customer\_id) references customer(customer\_id)  
    on delete no action on update no action;
- This constraint is to create a relationship between customer and address table.



- alter table phone\_number  
add constraint FK\_CustomerPhone  
foreign key (customer\_id) references customer(customer\_id)  
on delete no action on update no action;
- This constraint is to create a relationship between customer and phone number table.
  
- alter table product  
add constraint FK\_ProductCategory  
foreign key (category\_id) references category(category\_id)  
on delete no action on update no action;
- This constraint is to create a relationship between product and category table.
  
- alter table cart  
add constraint FK\_CartCustomer  
foreign key (customer\_id) references customer(customer\_id)  
on delete no action on update no action;
- This constraint is to create a relationship between cart and customer table.
  
- alter table cart  
add constraint FK\_CartProduct  
foreign key (product\_id) references product(product\_id)  
on delete no action on update no action;
- This constraint is to create a relationship between cart and product table.
  
- alter table comment  
add constraint FK\_CommentCustomer  
foreign key (customer\_id) references customer(customer\_id)  
on delete no action on update no action;
- This constraint is to create a relationship between comment and customer table.
  
- alter table comment  
add constraint FK\_CommentProduct

foreign key (product\_id) references product(product\_id)

on delete no action on update no action;

- This constraint is to create a relationship between comment and product table.

- alter table order\_table

add constraint FK\_Order\_tableCart

foreign key (cart\_id) references cart(cart\_id)

on delete cascade on update no action;

- This constraint is to create a relationship between order and cart table.

- alter table order\_table

add constraint FK\_Order\_tableAddress

foreign key (address\_id) references address(address\_id)

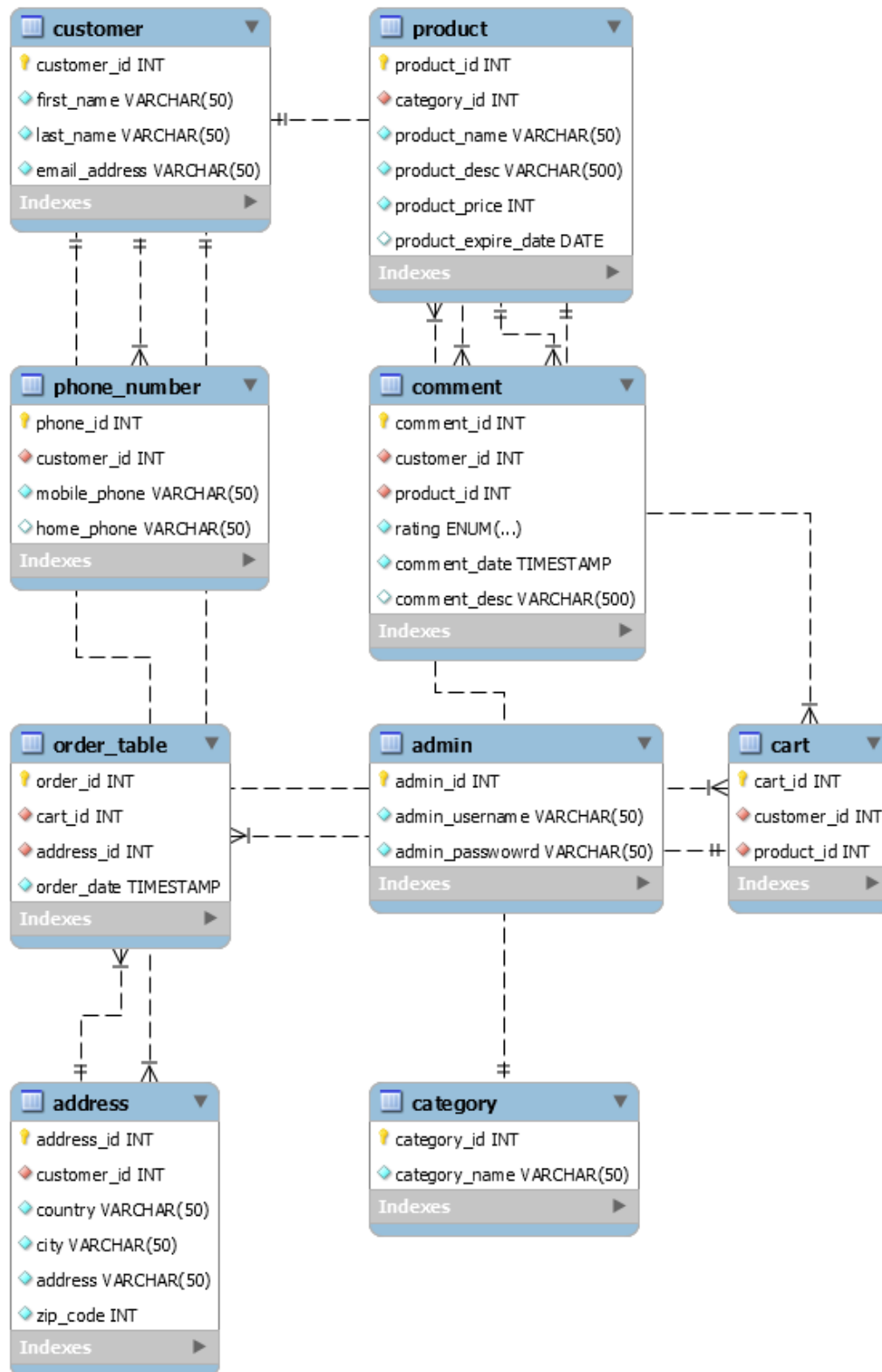
on delete no action on update no action;

- This constraint is to create a relationship between order and address table.

### **Additional Scripts**

- insert into customer values(NULL, 'yilmaz', 'sanli', '[yilmaz.sanli@agu.edu.tr](mailto:yilmaz.sanli@agu.edu.tr)');
- insert into address values(NULL, '1', 'Turkey', 'Kayseri', 'Agu Sumer Campus', 38010);
- insert into phone\_number values(NULL, '1', '05555555555', NULL);
- insert into category values(NULL, 'Electronic');
- insert into product values (NULL, 1, 'Laptop', '4 core computer', 2500, NULL);
- insert into cart values(NULL, 1, 1);
- insert into comment values(NULL, 1, 1, 'Excellent', '2020-05-09 12:00:08', 'The laptop is very fast. I recommend this laptop!');
- insert into admin values(NULL, 'admin', 'e10adc3949ba59abbe56e057f20f883e');
- insert into order\_table values(NULL, 1, 2, '2020-04-12 12:22:08');

### MySQL Workbench Generated E-R Diagram



# Datasets

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

logindb  
new\_schema  
projectdb  
sakila  
shop\_db

Tables

address  
admin  
cart  
category  
comment  
customer  
order\_table  
phone\_number  
product

Views  
Stored Procedures  
Functions

Administration Schemas

Information

Table: address

Columns:

address\_id int AI PK  
customer\_id int  
country varchar(50)  
city varchar(50)  
address varchar(50)  
zip\_code int

Query 1 address address

Limit to 1000 rows

1 • SELECT \* FROM shop\_db.address;

Result Grid

| address_id | customer_id | country | city    | address          | zip_code |
|------------|-------------|---------|---------|------------------|----------|
| 1          | 1           | Turkey  | Kayseri | Agu Sumer Campus | 38010    |

Output

Address 1 x

Apply Revert Context Help Snippets

Action Output

| # | Time     | Action                                      | Message           | Duration / Fetch      |
|---|----------|---------------------------------------------|-------------------|-----------------------|
| 1 | 07:44:59 | SELECT * FROM shop_db.address LIMIT 0, 1000 | 0 row(s) returned | 0.000 sec / 0.000 sec |
| 2 | 07:46:38 | SELECT * FROM shop_db.address LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |

Object Info Session

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

MySQL Workbench

Local instance MySQL80

File Edit View Query Database Server Tools Scripting Help

Navigator

SCHEMAS

Filter objects

logindb  
new\_schema  
projectdb  
sakila  
shop\_db

Tables

address  
admin  
cart  
category  
comment  
customer  
order\_table  
phone\_number  
product

Views  
Stored Procedures  
Functions

Administration Schemas

Information

Table: product

Columns:

product\_id int AI PK  
category\_id int  
product\_name varchar  
product\_desc varchar  
product\_price int  
product\_expire\_date date

Query 1 address address product

Limit to 1000 rows

1 • SELECT \* FROM shop\_db.product;

Result Grid

| product_id | category_id | product_name | product_desc    | product_price | product_expire_date |
|------------|-------------|--------------|-----------------|---------------|---------------------|
| 1          | 1           | Laptop       | 4 core computer | 2500          | NULL                |

Output

product 1 x

Apply Revert Context Help Snippets

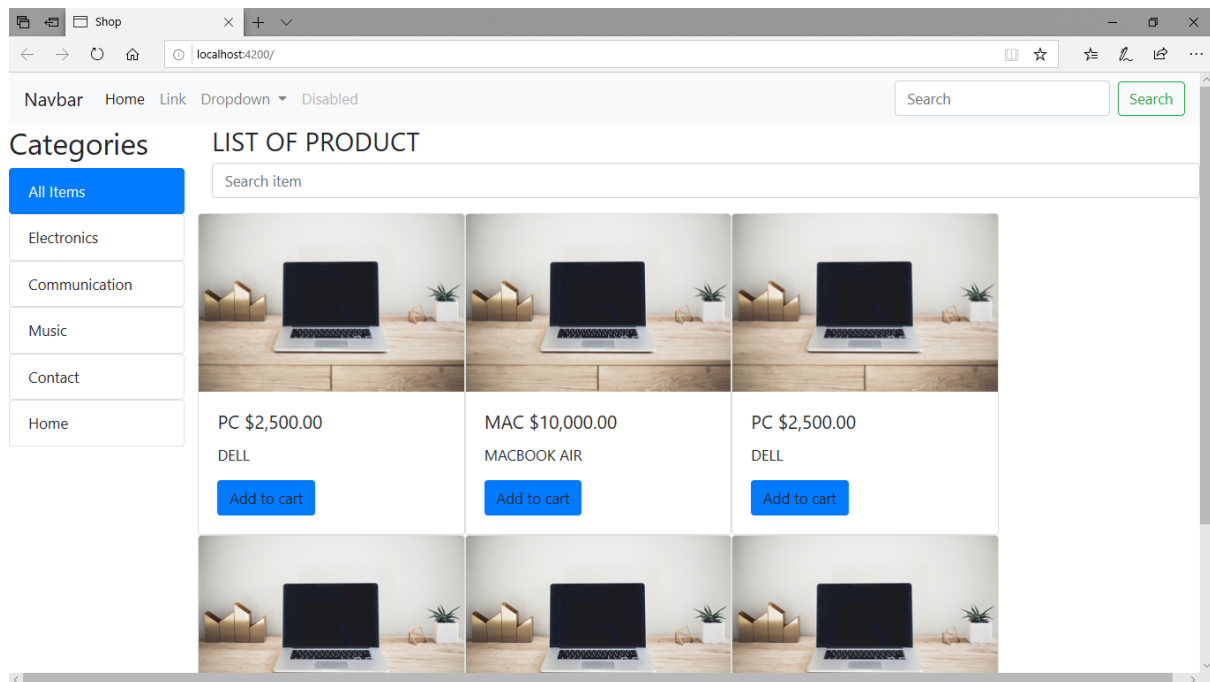
Action Output

| # | Time     | Action                                      | Message           | Duration / Fetch      |
|---|----------|---------------------------------------------|-------------------|-----------------------|
| 1 | 07:44:59 | SELECT * FROM shop_db.address LIMIT 0, 1000 | 0 row(s) returned | 0.000 sec / 0.000 sec |
| 2 | 07:46:38 | SELECT * FROM shop_db.address LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |
| 3 | 07:47:26 | SELECT * FROM shop_db.product LIMIT 0, 1000 | 1 row(s) returned | 0.000 sec / 0.000 sec |

Object Info Session

Automatic context help is disabled. Use the toolbar to manually get help for the current caret position or to toggle automatic help.

## Demo Application



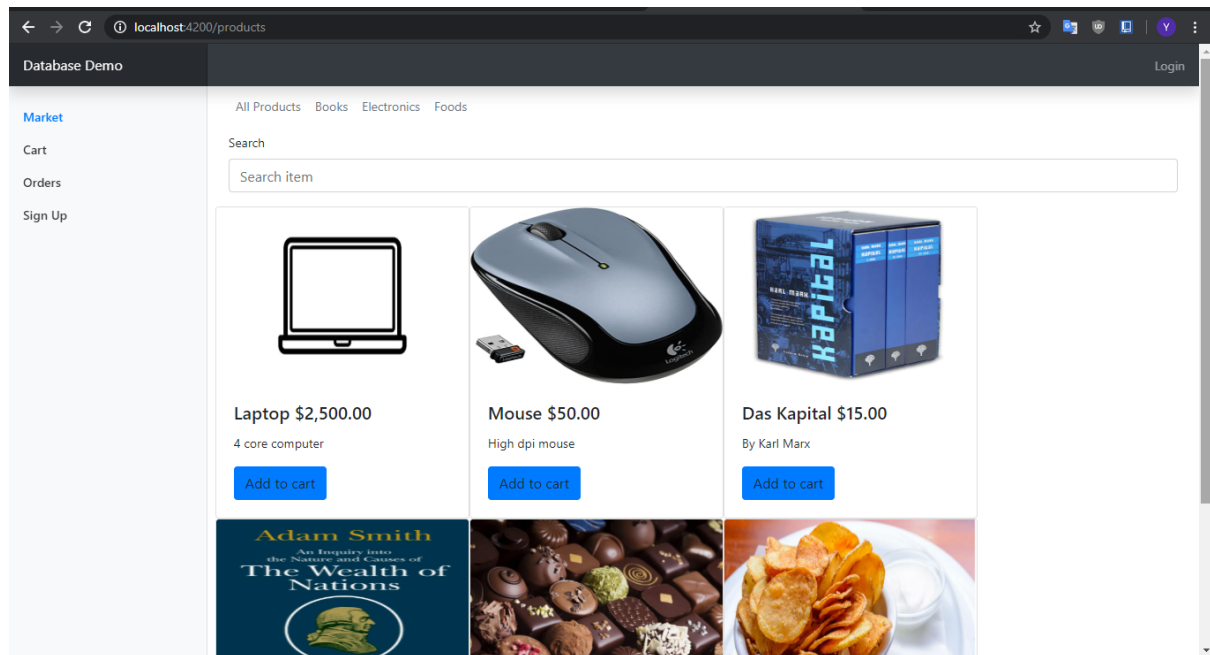
The example design of the shop application.

## Last Version of the Demo Application

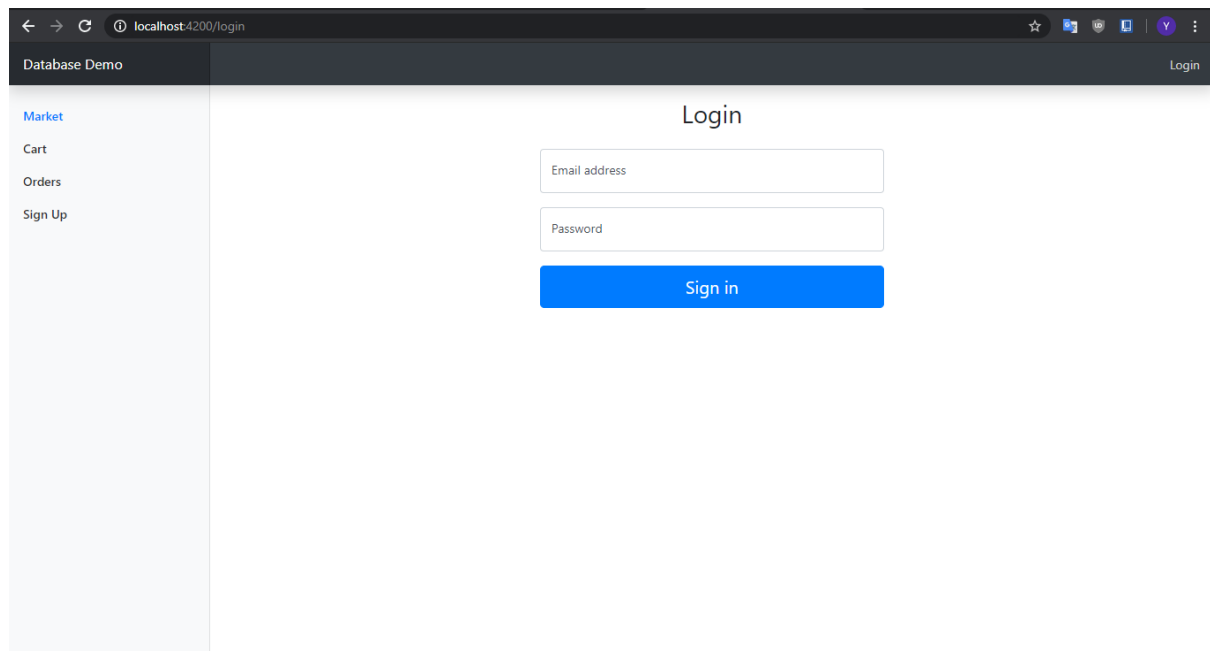
In the last version of the application, we have improved the user interface. In this system, a user can register the system by using sign up service. After registration, user can enter the system to add an item to their cart. The user can see all the items, filter items according to the category and search for specific item. In addition, user can add items to the cart.

The demo application is developed by using Spring Boot and Angular. In the backend of the website, we have developed a REST API which communicates with MySQL to get the data and returns the data in JSON format. API provides different data for different URLs. In addition, authentication service is also done by API. In the frontend of the website, a responsive user interface is designed and data which comes from API is presented in these user interfaces.

## Main Page



## Login Page



# Signup Page

Database Demo

Login

Market

Cart

Orders

Sign Up

Sign Up

Name

Surname

Email

Password

Country

City

Address

Zip\_code

Sign Up

# Cart Page



Database Demo

Logout

Market

Cart

Orders

| Product                                                                                                                                                                         | Price | Action |
|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|--------|
| <div><div></div><div>Laptop</div></div> <div>\$2,500.00</div> <div><div>x Remove</div></div> |       |        |
| <div><div></div><div>Mouse</div></div> <div>\$50.00</div> <div><div>x Remove</div></div>     |       |        |

Order