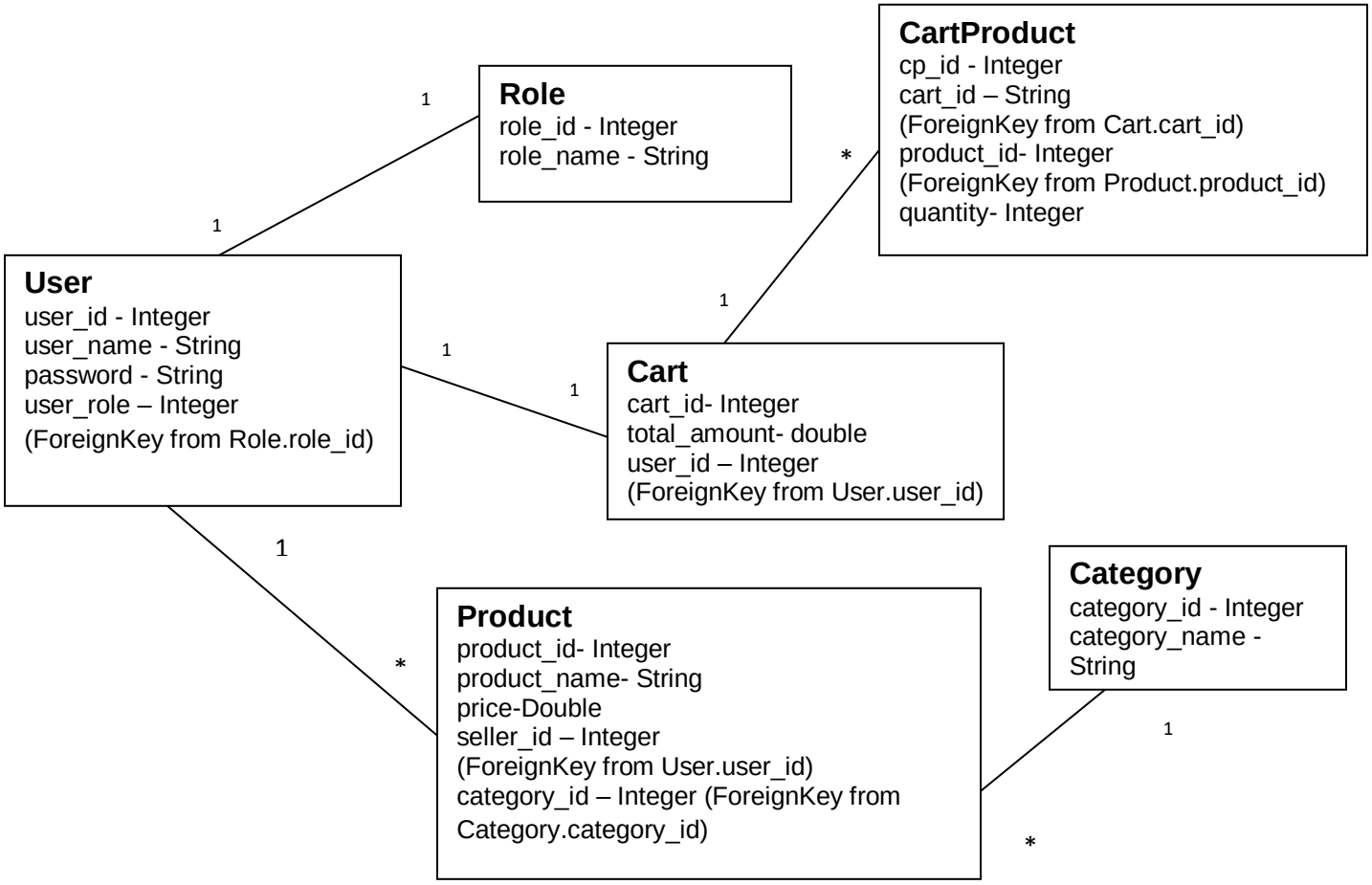
McDiffyStore is a vendor who sell various products ranging across different categories. They hired you to develop a distributed e-commerce platform to move their business online. As an initial MVP you are required to develop a restful API backend application in FLASK.

Here is the requirement for the application.



Database models are already created and initialized as below:

DB initialised with following default data:

Category	**Role**	**User**	**Cart**
category_id, **category_name** 'Fashion' 'Electronics' 'Books' 'Groceries' 'Medicines'	**role_id,role_name** 1,'CONSUMER' 2,'SELLER'	**user_id,user_name,** **password,user_role** 1,'jack', 'pass_word',1 2,'bob', 'pass_word',1 3,'apple', 'pass_word',2 4,'glaxo', 'pass_word,2	**total_amount,** **user_id** 20, 1 0, 2

Product	**CartProduct**
product_id,price, product_name, category_id, **seller_id** 1, 29190, 'ipad', 2, 3 10, 'crocin', 5, 4	**cp_id, cart_id, product_id, quantity** 1,1, 2, 2

Your job is to create the following APIs, use JWT authentication with roles to protect consumer and seller specific endpoints. The JWT is included in the header with key **JWT** and value as jwt token.

- Consumers can search, add, update and delete items in cart.
- Sellers can add, update and delete products to the database.
- APIs preceeding with /api/public are public APIs and can be accessed by anyone.
- APIs preceeding with /api/auth/consumer are authenticated and consumer APIs.
- APIs preceeding with /api/auth/seller are authenticated and seller APIs
- if authenticated endpoints are accessed without JWT, return 401.
- if a consumer endpoint is accessed with seller JWT or vice versa, return 403.

Below are public endpoints:

/api/public/product/search –GET - This endpoint takes a query parameter 'keyword' and returns all the matching products containing the keyword in productName with the category details. **Hint** :Perform Join	
Example: */api/public/product/search?keyword="crocin"*	*[* *{* *"category": {* *"category_id": 5,* *"category_name": "Medicines"* *},* *"price": 10.0,* *"product_id": 2,* *"product_name": "crocin",* *"seller_id": 4* *}* *]* ***status** 200*

/api/public/product/search?keyword ="paracetamol"	**status**400

/api/public/login - POST - takes username and password in json body, authenticates and returns JWT. Can authenticate both consumer and seller.	
request body – *{"username": "bob","password": "pass_word"}*	**status***200*, **response body** – *eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMeKKF2QT4fwpMeJf36POk6yJV_adQssw5c*
request body – *{"username": "bob","password": "password"}*	**status** 401

<u>Below are all authenticated endpoints:</u>

CONSUMER ENDPOINTS

Logged in as Jack

/api/auth/consumer/cart- *<u>GET</u>*- returns the consumer's cart with all details unfolded. *Hint: Perform Join on Cart,CartProduct,Product,Category*	
/api/auth/consumer/cart	**status***200*, **response body** *– [* *{* *"cartproducts": {* *"product": {* *"product_id": 2,* *"price": 10.0,* *"product_name": "crocin",* *"category": {* *"category_name": "Medicines",* *"category_id": 5* *}* *},* *"cp_id": 1* *},* *"cart_id": 1,* *"total_amount": 20.0* *}* *]*

/api/auth/consumer/cart- *POST*– takes a product id and quantity in request body and adds it to the consumer's cart and returns the total amount of the cart.	
Request body- *{"product_id":1,"quantity":1}*	**status** *200* **Response body** 29210.0 *Explanation:* [Product 1 (added now)and Product 2 (already there by default as shown in database) added to cart of user jack]
If product already in cart is added again *{"product_id":2,"quantity":1}*	**status** 409

/api/auth/consumer/cart- *PUT*– takes a product id and quantity in request body and updates the quantity of the product in the consumer's cart and returns the updated total amount of the cart.	
Request body- *{"product_id":1,"quantity":2}*	**status** *200* **Response body** 58400.0 *Explanation:* [Product 1 of two quantities (updated now) and Product 2 of two quantities (already there by default as shown in database) added to cart of user jack]

/api/auth/consumer/cart- *DELETE*– takes a Product id in request body and removes the product from the cart. Return the final total amount of the user cart	
Request body- *{"product_id":1}*	**status** *200* ***Response Body*** *20.0* *Explanation:* [Product 2 of two quantities (already there by default as shown in database) is present in cart of jack after deletion]

SELLER ENDPOINTS

Logged in as Apple

/api/auth/seller/product- *GET*– return all the products owned by seller. ***/api/auth/seller/product/{productId}-*** *GET*– return the product identified by the supplied path parameter productId.If product not owned by seller return 404	
/api/auth/seller/product	**status***200,* ***response body****-* *[* *{* *"category": {* *"category_id": 2,* *"category_name": "Electronics"* *},* *"price": 29190.0,* *"product_id": 1,* *"product_name": "ipad",* *"seller_id": 3* *}* *]*
/api/auth/seller/product/1	**status** 200**,** **response body -** *[{* *"category": {* *"category_id": 2,* *"category_name": "Electronics"* *},* *"price": 29190.0,* *"product_id": 1,* *"product_name": "ipad",* *"seller_id": 3* *}]*
/api/auth/seller/product/1 *Logged in as glaxo* *(product_id 1 is not his product)*	**status** 404

/api/auth/seller/product- *POST*– takes a product json in request body and saves it to database. Returns the newly added product id. *Note :If the product id is already present in table return status 409*	

Request body-
{
"product_id":3,
"product_name":"phone",
"price":80000,
"category_id":2
}
	status*201,*
	response : *3*

/api/auth/seller/product- *PUT*– takes a product_id and price in request body and updates the price of the respective product_id. *Note: If product not owned by seller return 404*	
Request body- *{"product_id":3,"price":8768678}*	**status***200*
Request body- *{"productId":3,"price":"986565.0"}* *Logged in as glaxo* *product_id 3 is not his product*	**status***404*

/api/auth/seller/product/{prodid}- *DELETE*– takes a product id path parameter and deletes the product from the database. *Note: If product not owned by seller return 404*	
/api/auth/seller/product/3	**status***200*
/api/auth/seller/product/3 *Logged in as glaxo (product_id 3 is not his product)*	**status** 404

Take a look at the testcases to understand more on how the validation works.
Good Luck and Start Coding!

Run the application

Note: Run all the commands for the application from the directory where main.py file is located.

1. export FLASK_APP=main
2. python3 -m flask db init (ignore the error→migrations folder not empty)
3. python3 -m flask db migrate
4. python3 -m flask db upgrade
5. python3 -m flask run

Test the application

Note: Run the test command for the application from the directory where tests.py file is located.

1. python3 -m pytest tests.py

SQLITE3 commands

Note: Go to the folder where your .db files are present

1. *'sqlite3'* command will get you into sqlite3 terminal.
2. *'.open dbname.db'* will get that database as core. All further operations will be made on that database.
3. *'.tables'* will list the tables in that database.
4. *'select * from tablename; '* will list the rows of table.

Note: All the operations in the table can be done with the similar commands as SQL