Creating APIs for Coffee House

https:/coffee.com

Who will be the users of the APIs?

Owner(admin), Baristas, Customers

What data points does the Coffee House have?

Will Be Using

- **Users**: These are the customers who visit the coffee house and use its services. User data might include information like name, contact details, order history, and payment methods.
- **Administrator**: This represents the staff members who manage the coffee house operations. Administrators would have different permissions compared to regular users, such as the ability to add or remove products, manage orders, and view sales reports.
- **Menu**: This represents the list of products that the coffee house offers. Each item on the menu would have related information such as the name, description, price, and possibly an image.
- **Products**: These are the individual items that make up the menu. Each product would have related information such as its ingredients, preparation time, cost, and availability status.
- **Orders**: This represents the orders placed by customers. Each order would have related information such as the items ordered, the total cost, the time of the order, and the status of the order (e.g., pending, preparing, ready, delivered).
- **Info**: Basic information about coffee shop - Location, Baristas, About, Mission statement etc.
- **Table Reservations**: If the coffee house accepts reservations, information about table bookings.
- **Reviews**: Customer reviews and ratings for products or the coffee house itself.

Will Not Be Using In Next Steps but could be used

- **Delivery Information**: If the coffee house offers delivery, information about delivery zones, fees, etc.
- **Inventory**: Information about the stock of products and ingredients.
- **Locations**: If the coffee house has multiple locations, information about each location.
- **Promotions**: Information about any ongoing or upcoming promotions or discounts.
- **Loyalty Program**: Information about a customer loyalty program, if one exists.

- **Suppliers**: Information about the suppliers for the coffee house.
- **Employees**: Information about the employees, beyond just the administrators.
- **Sales Reports**: Information about sales, useful for administrators for tracking business performance.
- **Payment Methods**: Information about how customers can pay for their orders.
- **Order Status**: Information about the status of an order (e.g., pending, preparing, ready, delivered).

1st part - creating User and Product models

## What are the data attributes associated with a user in the Coffee House application?

User

field	unique	type	automatically assigned by computer	default value
userId	true	int	true	-
username	false	string	false	-
email	true	string	false	-
password	false	string	false	-

Example

userId	username	email	password
1	maris	maris@example.com	secret

Group Work (8minutes)

- **What could be data attributes associated with a product in the Coffee House application?**

Product

field	unique	type	automatically assigned by computer	default value

2nd part - creating routes

What actions need to be performed on selected data points? What kind of security does the API need(access rights)?

@desc (description)

```
C - Create (POST)
R - Read (GET)
U - Update [PUT - update everything | PATCH - partially update something]
D - Delete (DELETE)
```

@route

```
METHOD [POST, GET, PUT, PATCH, DELETE etc.] and path after baseUrl (https:/coffee.com) [/api/v1,
```

@access

```
Public - Everyone can access
Private - Only Users can access
Private/Admin - Only Users with admin right can access
```

@desc	@route	@access
Create New User	POST /api/v1/auth/register	Public
Get All Users	GET /api/v1/users	Private/Admin
Get Single User	GET /api/v1/users/:userId	Private/Admin
Update Users Password	PATCH /api/v1/users/	Private

@desc	@route	@access
Delete A User	DELETE /api/v1/users/:userId	Private/Admin

Group Work (8minutes)

- **Make similar table for Orders API**

3rd part - response or error

How will the API handle errors and communicate these to the client?

@desc	@route	@access	@throwError
Create New User	POST /api/v1/users	Public	email exists; server crashes
Get All Users	GET /api/v1/users	Private/Admin	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes
Get Single User	GET /api/v1/users/:userId	Private/Admin	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist
Update Users Password	PATCH /api/v1/users/:userId	Private	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist; password and

@desc	@route	@access	@throwError
			confirmPassword fields in body does not match
Delete A User	DELETE /api/v1/users/:userId	Private/Admin	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist

Public or Private user tries to access admin APIs?
Computer(Server) crashes?
Passwords don't match?

It is good practice to give appropriate status code when we send response, but a lot of time it is hard to distinct which one to use! It is okay not to pick perfect match.

Code	Constant	Reason Phrase	Description
204	NO_CONTENT	No Content	The server successfully processed the request and is not returning any content.
400	BAD_REQUEST	Bad Request	The server could not understand the request due to invalid syntax.
401	UNAUTHORIZED	Unauthorized	The request requires user authentication.
403	FORBIDDEN	Forbidden	The server understood the request, but it refuses to authorize it.
404	NOT_FOUND	Not Found	The server can't find the requested resource.
405	METHOD_NOT_ALLOWED	Method Not Allowed	The method specified in the request is not allowed for the resource identified by the request URI.

Code	Constant	Reason Phrase	Description
500	INTERNAL_SERVER_ERROR	Internal Server Error	The server encountered an unexpected condition which prevented it from fulfilling the request.
502	BAD_GATEWAY	Bad Gateway	The server was acting as a gateway or proxy and received an invalid response from the upstream server.
503	SERVICE_UNAVAILABLE	Service Unavailable	The server is currently unable to handle the request due to a temporary overloading or maintenance of the server.

If everything is okay - what we want to send to the client?

Success Message
Data
Old Data
Updated Data
Deleted Data
Selection of the Data (e.g. crucial not to send password with users information)
Combination of Success Message and Data
Metadata - advanced

@desc	@route	@access	@throwError	@OK
Create New User	POST /api/v1/users	Public	email exists; server crashes	json with message - user created and data without password

@desc	@route	@access	@throwError	@OK
Get All Users	GET /api/v1/users	Private/Admin	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes	json with all users data
Get Single User	GET /api/v1/users/:userId	Private/Admin	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist	json with single users data
Update Users Password	PATCH /api/v1/users/:userId	Private	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist; password and confirmPassword fields in body does not match	json with message - user updated successfully and new users data
Delete A User	DELETE /api/v1/users/:userId	Private/Admin	need header 'api_username': "username" and	json with message -

@desc	@route	@access	@throwError	@OK
			"api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist	user deleted successfully

What kind of rate limiting (if any) will the API need?

This is to prevent abuse of the API.

50 requests in a minute | 150 in 15 minutes

Group Work (8minutes)

- **Select two routes of Product API**
- **think of cases when we would need to refuse (send Error) when someone is using these routes?**
- **how would you respond if everything is OK?**
- **What would you set as rate limit to these routes**

P.S.! APIs need to be Asynchronous (async)!

it means that computer(server) can handle multiple actions at once. By default a lot of programming languages are synchronous - one action at a time.

Image if Amazon had Synchronous APIs - In real world it would be like having entire Latvia's population standing in line for 1 cashier to handle orders.

Utilizing Copilot (GPT4)

PUBLIC CODE ALLOWED

Please Create Flask with all routes + SQLite project (coffee_house.db) every route needs to be async
it is demo project for training, could you avoid password hashing
authentication will be sent through headers api_username and api_key for Protected routes.

Protected/Admin routes should have additional header 'who_is_the_boss': "IAMTHEBOSS";
invoke error if some @throwError column criteria is true
could you please add try catch blocks to all routes

Users schema:

data will be sent in request body.
field unique type automatically assigned by computer default value
userId true int true -
username false string false -
email true string false -
password false string false -

User Routes And Controller Setup:

before each route in code, please document it accordingly like this, IMPORTANT!:

```
# @desc Create New User
# @route POST /api/v1/users
# @access Public
def create_user():
    try:
      ....
    except Exception as e:
        return jsonify(error=str(e)), 400
```

@desc	@route	@access	@throwError	@OK
Create New User	POST /api/v1/users	Public	email exists; server crashes	json with message - user created and data without password
Get All Users	GET /api/v1/users	Private/Admin	need header 'api_username':	json with all users data

@desc	@route	@access	@throwError	@OK
			"username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes	
Get Single User	GET /api/v1/users/:userId	Private/Admin	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist	json with single users data
Update Users Password	PATCH /api/v1/users/:userId	Private	need header 'api_username': "username" and "api_key":'password' and both matches db entries; needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist; password and confirmPassword fields in body does not match	json with message - user updated successfully and new users data
Delete A User	DELETE /api/v1/users/:userId	Private/Admin	need header 'api_username': "username" and "api_key":'password' and both matches db entries;	json with message - user deleted successfully

@desc	@route	@access	@throwError	@OK
			needs header 'who_is_the_boss': "IAMTHEBOSS"; if server crashes; userId does not exist	

Please use these formulas to authenticate protected routes accordingly

```
def authenticate(username, password):
user = User.query.filter_by(username=username, password=password).first()
if user is None:
abort(401)
return user

def authenticateAdmin(who_is_the_boss):
if who_is_the_boss != 'IAMTHEBOSS':
abort(401)
return True
```