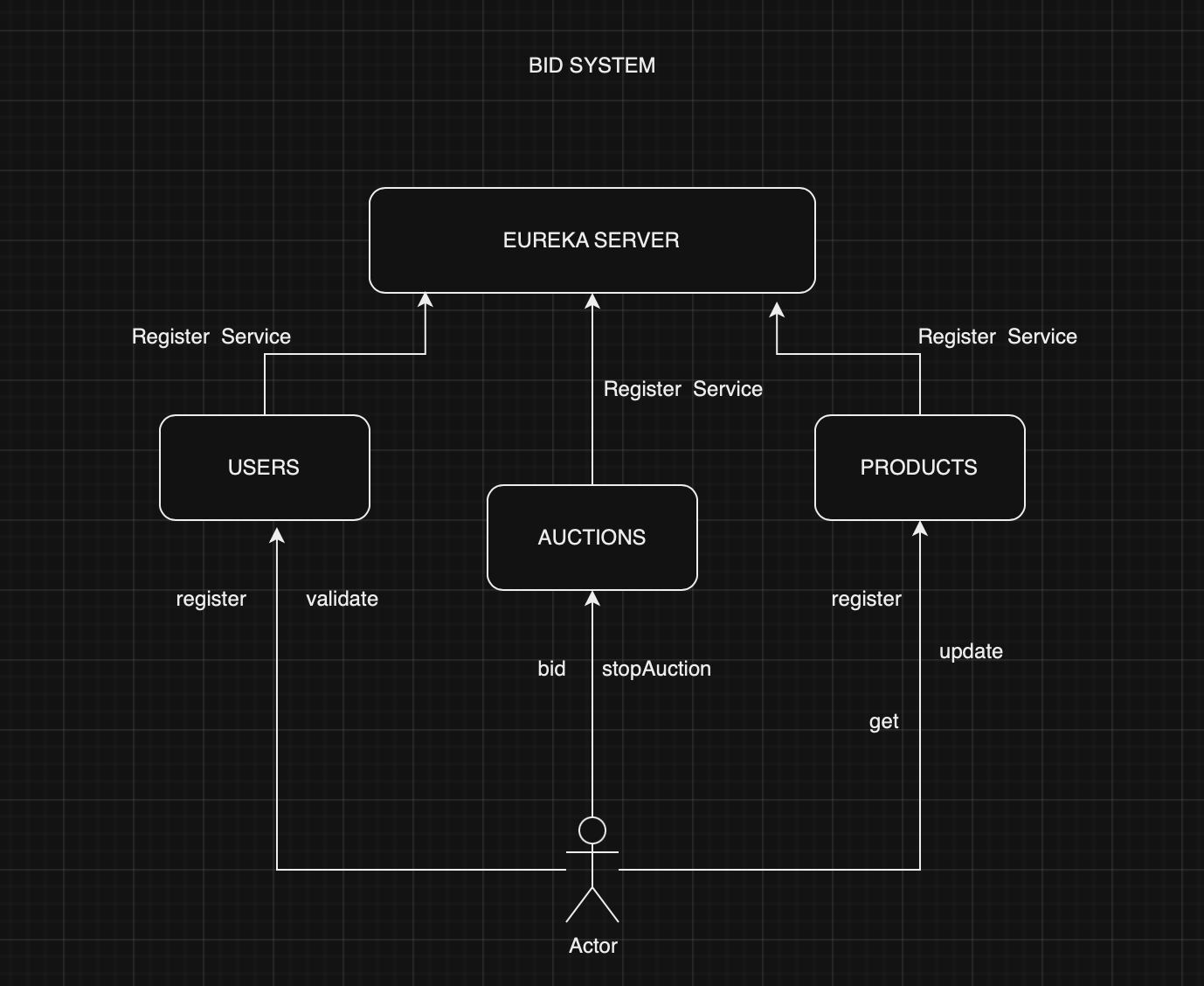
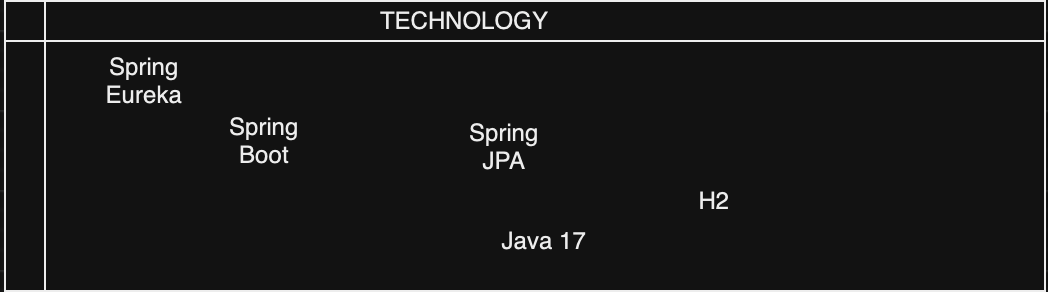
# BID SYSTEM

## High Level Architecture:





## Minimum Viable Product

The Bidding System is based on Microservices Architecture and has 3 Microservices and a Service Discovery alongwith it.

* Eurekaserver
* Users
* Products
* Auctions

A brief description of all the services is mentioned in the following sections.

## Services Description:

EurekaServer:

This is a Service discovery project and used to register the services so that any new Microservices introduced or removed can be monitored and managed. All the services internally use this Server to call other services.

Users:

The sole purpose of this service is to manage users. It has 2 endpoints which are register and validate and as the name indicates, register is used to add new users to the system and validate checks if the users are good.

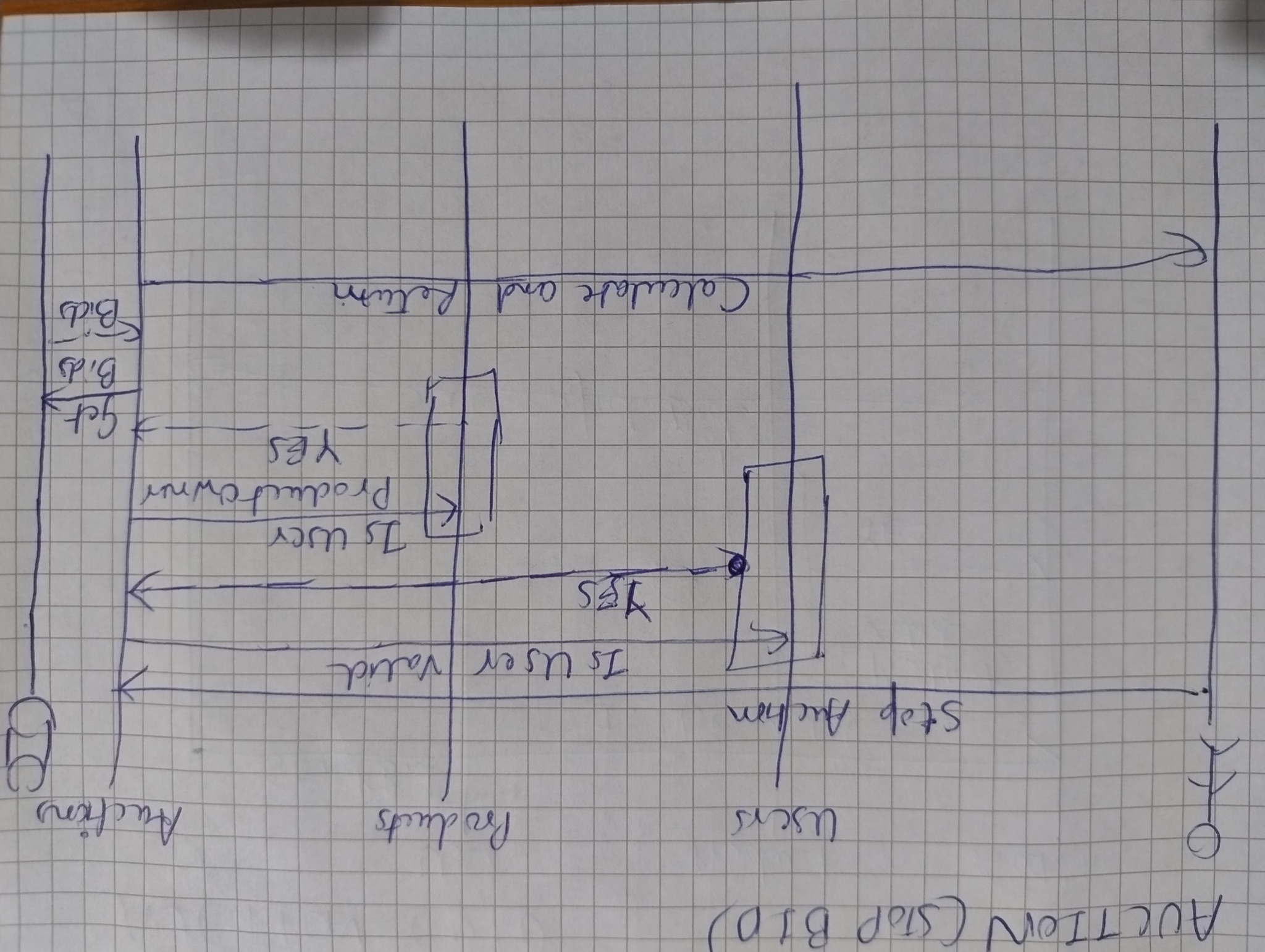
Products:

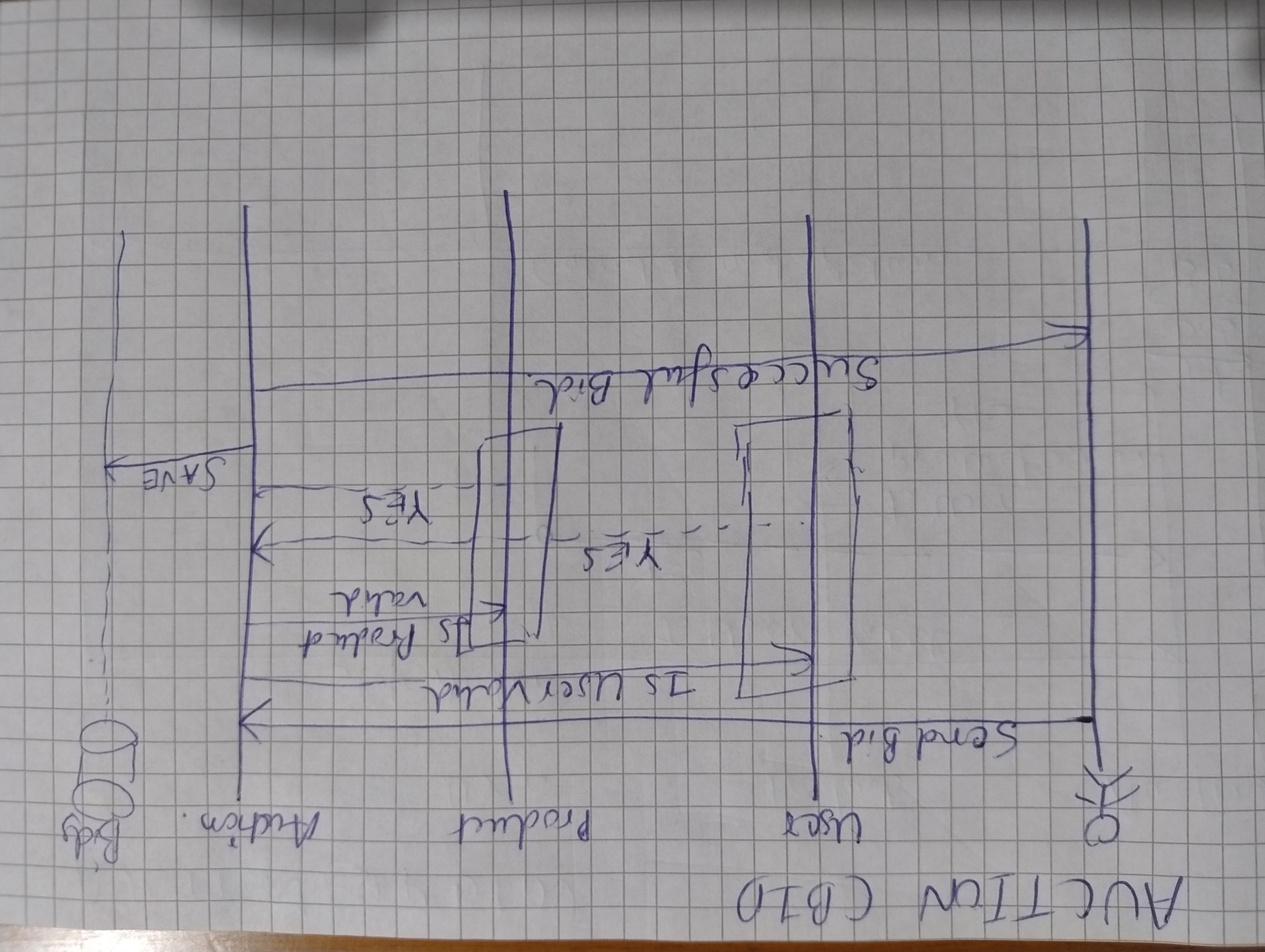
This Service is taking care of products in the system. It can do all the operations related to Products but before that it validates if the operating person is registered or not. It gets this information from users service.

Auctions:

This Service is involved in the bidding process. A user can use this service to bid for a product any number of times. But where are the products ? For getting and validating the products, this service takes help from PRODUCTS Microservice. Also, a valid user can only bid for the item and hence it is also dependent in USERS microservice for validating users.

## Sample Flow: Auctions





## Prerequisite:

Java 17 must be present on your system and preferably added to the path.

## 

## User Instructions:

1. Clone the projects in your local machine from below mentioned URLs
   1. <https://github.com/gaurravkumar/eurekaserver>
   2. <https://github.com/gaurravkumar/users>
   3. <https://github.com/gaurravkumar/products>
   4. <https://github.com/gaurravkumar/auctions>
2. Build the projects on your local machine using Maven. You can either use command line and do it using **mvn clean package** or use the IDE to build it to ensure it is building fine on your local machine. This command or building from IDE will create a jar file in **<PROJECT>/target** folder which can be referenced to start the projects.
3. Start the project from IDE start button at the top right corner or use the command java -jar <Location of the JAR created>
4. eurekaserver project must be started as the first project and other 3 (users, products and auctions) can be started in any order.

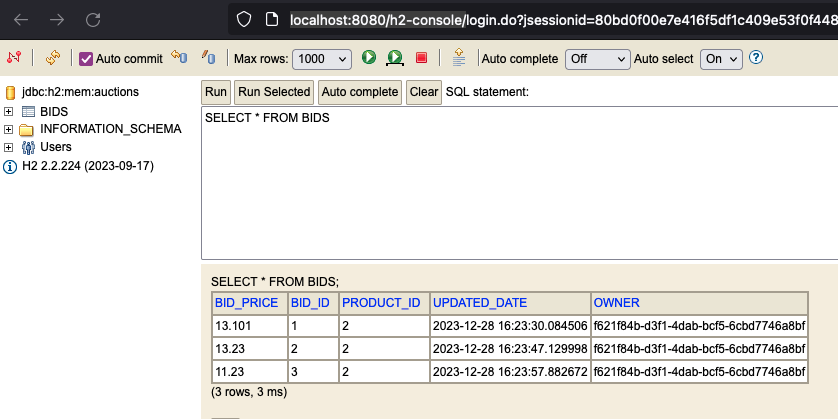
## Database View:

Each of the project have H2 console enabled and the information can be seen on these URLs

Users: <http://localhost:8082/h2-console/>

Products: <http://localhost:8081/h2-console/>

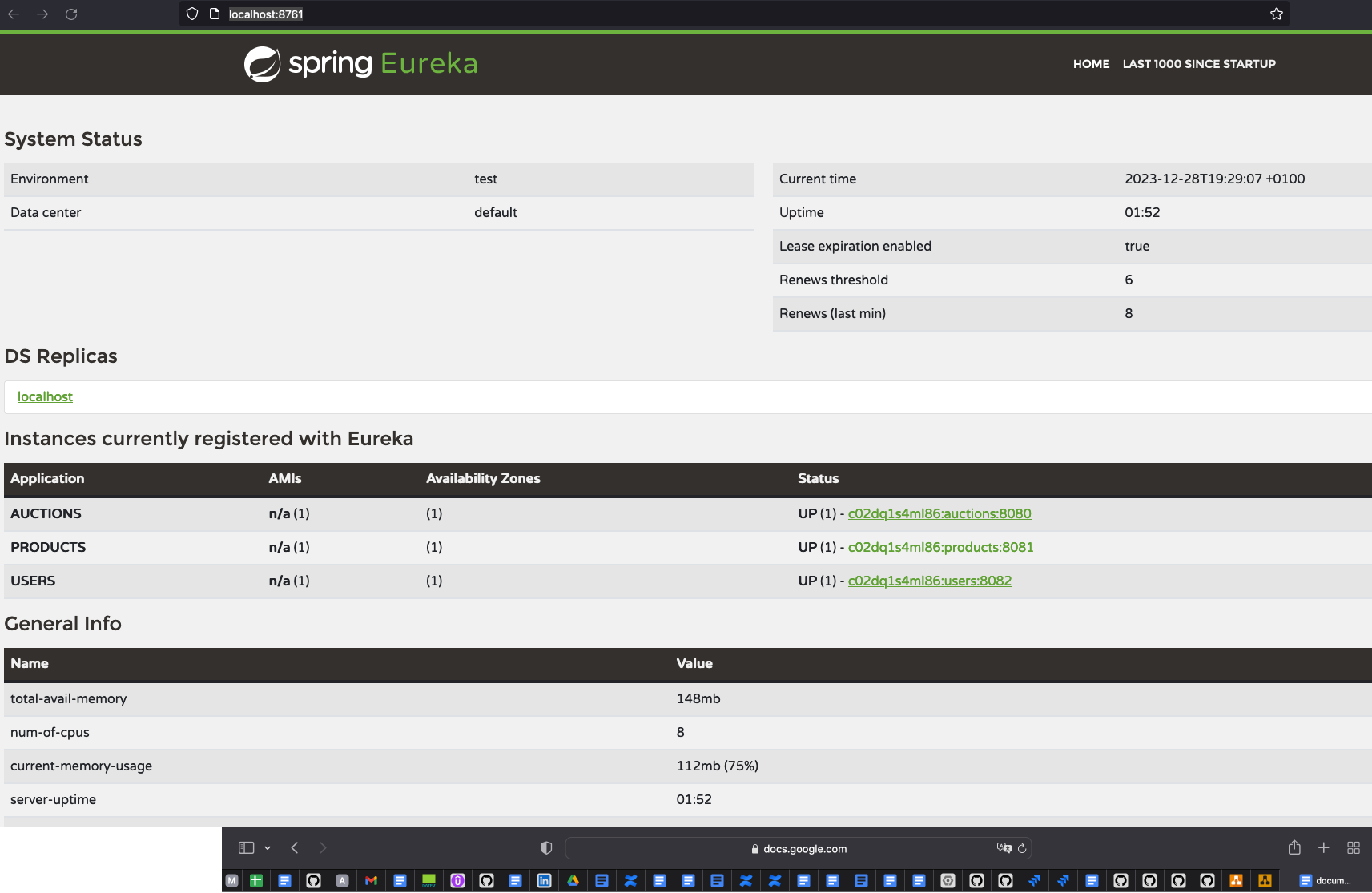
Auctions: <http://localhost:8080/h2-console/>



## Eureka Registry:

You can check whether your services got registered with eureka server by opening the following URL:

<http://localhost:8761/>



## Sample Requests: For All Services

Install Postman on your machine and try to make various requests using the following order:

1. Register the USER otherwise you will not be able to register product or make a bid. Make sure you note down the token received in response.
   1. URL : <http://localhost:8082/api/users/register>

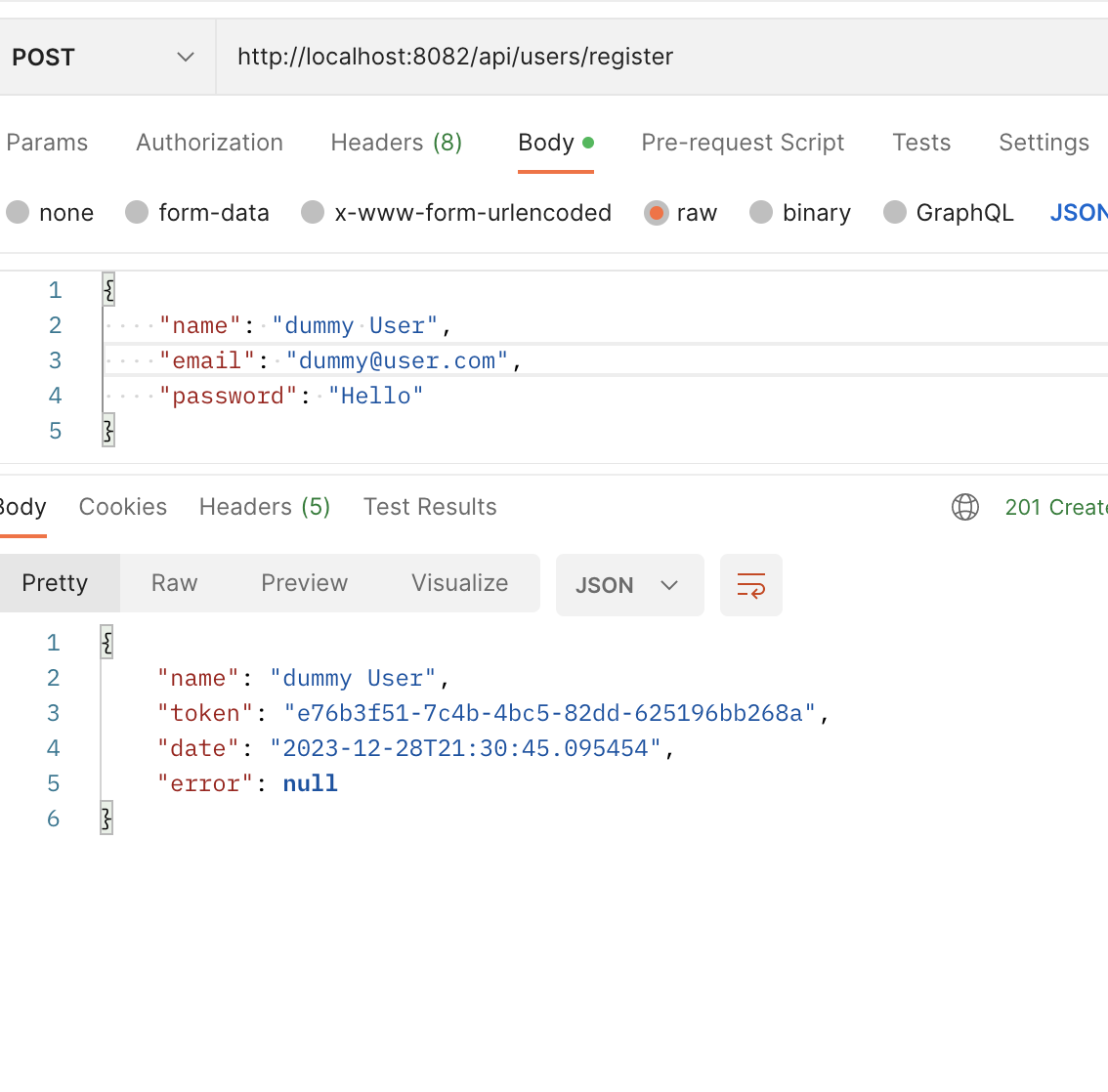
{

"name": "dummy User",

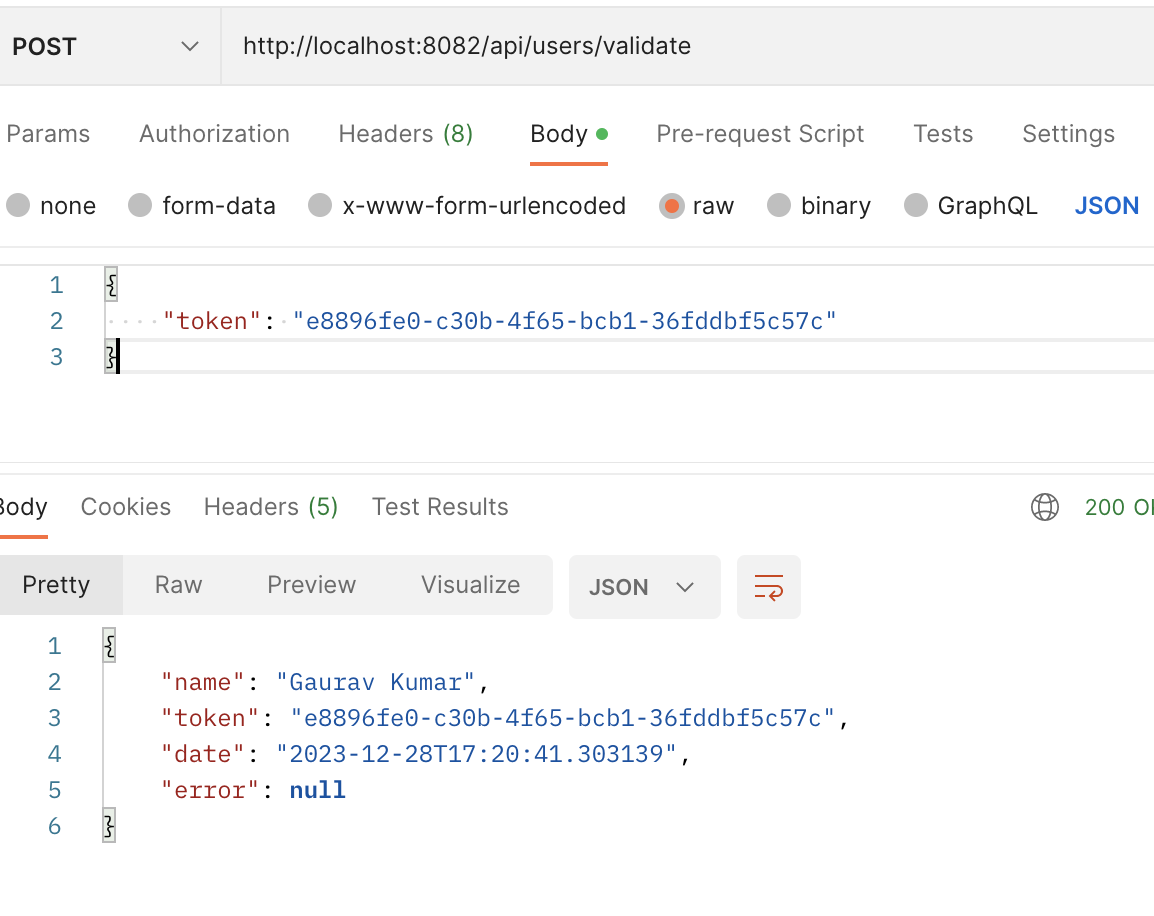
"email": "dummy@user.com",

"password": "Hello"

}



1. Validate user will be used internally but you can validate the user using postman.
   1. URL : <http://localhost:8082/api/users/validate>



1. After the user is registered you can use any of the following PRODUCTS URLs to register or view Products
   1. <http://localhost:8081/api/products/registerProduct>

Sample:**token:<VALUE>** should be added to headers

{

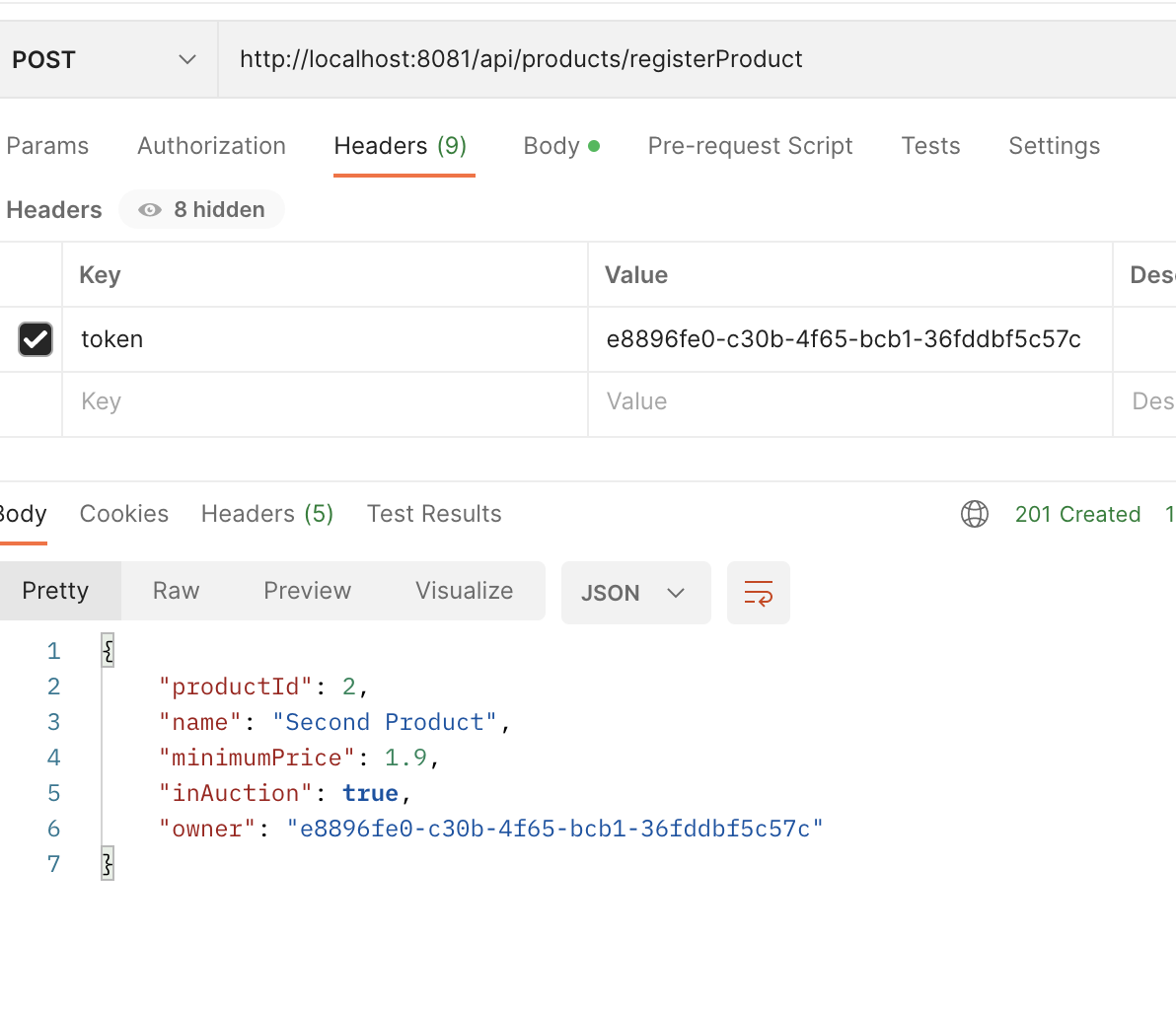
"name": "Second Product",

"minimumPrice": 1.9,

"inAuction": **true**

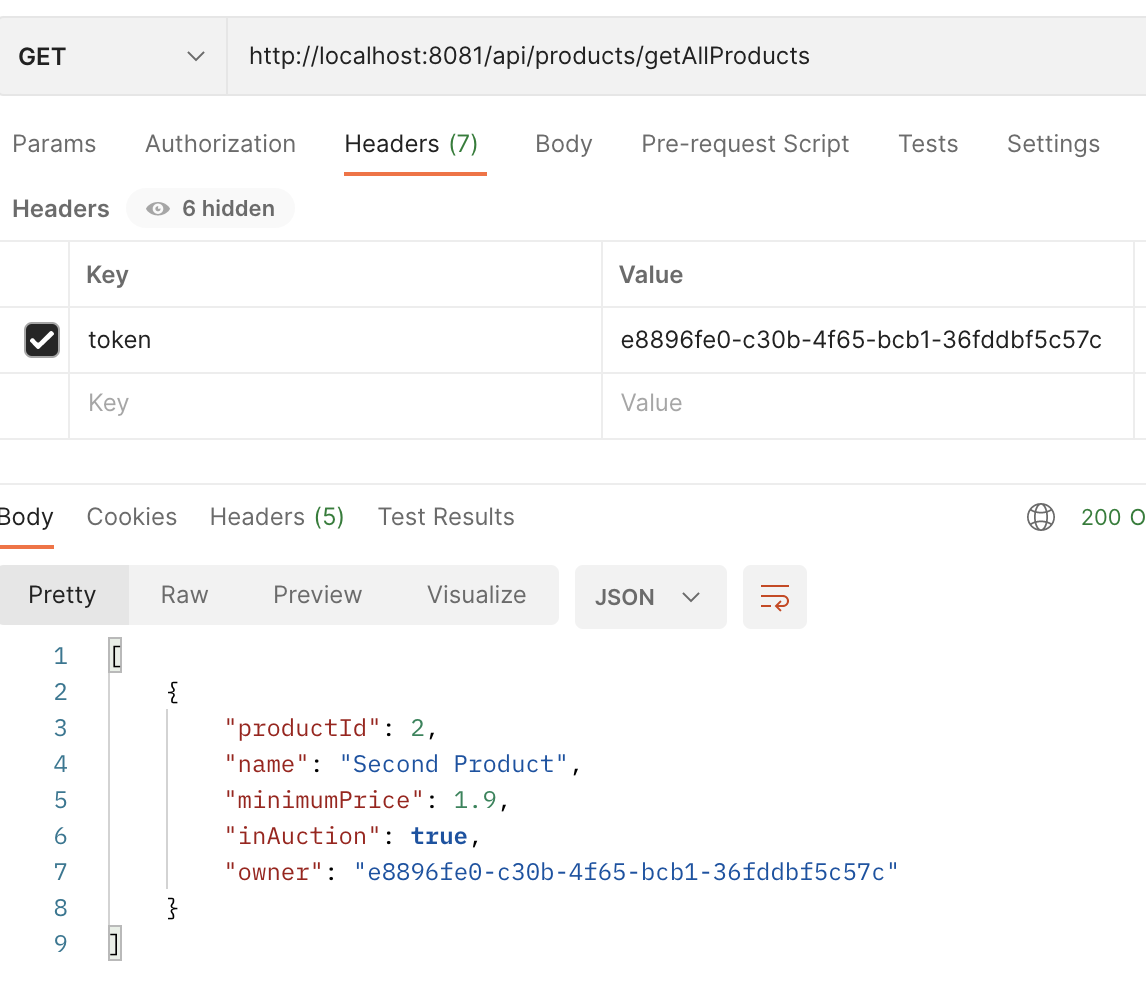
}

Make sure to enter token in headers as shown in screenshot below



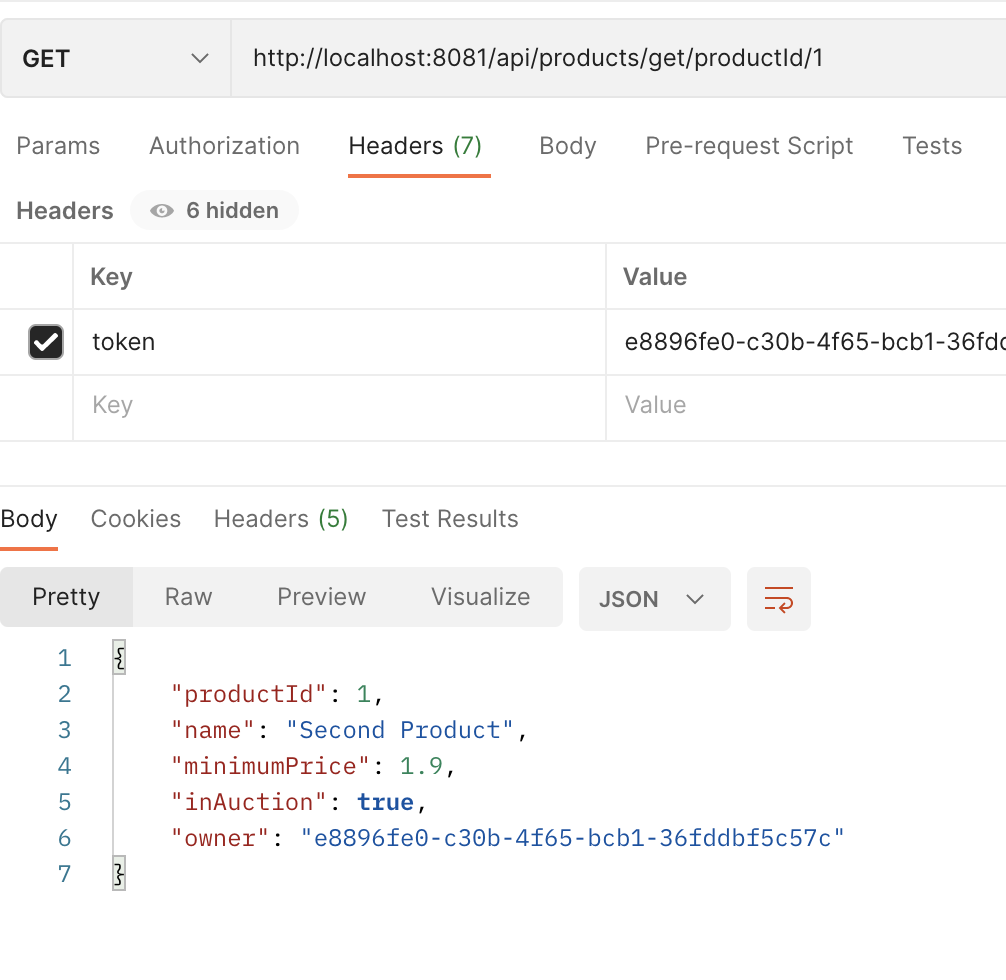
* 1. <http://localhost:8081/api/products/getAllProducts>

Sample:**token:<VALUE>** should be added to headers



* 1. <http://localhost:8081/api/products/get/productId/1>

Sample:



* 1. <http://localhost:8081/api/products/updateInAuctionStatus>

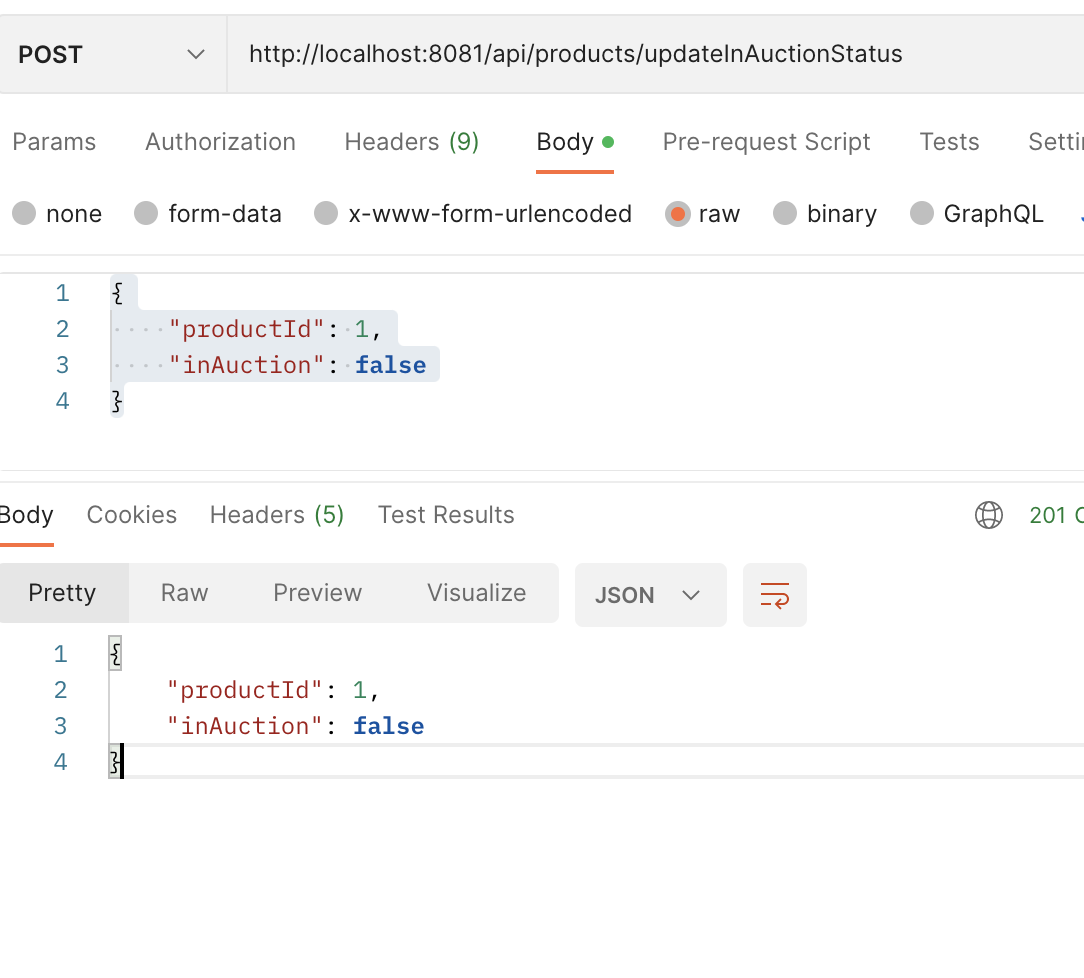
Sample: **token:<VALUE>** should be added to headers

{

"productId": 1,

"inAuction": **false**

}



1. Auctions can then be used to make bids. Make sure you note product Ids from Products API while registering. :
   1. URL for bids: <http://localhost:8080/api/auctions/bid>

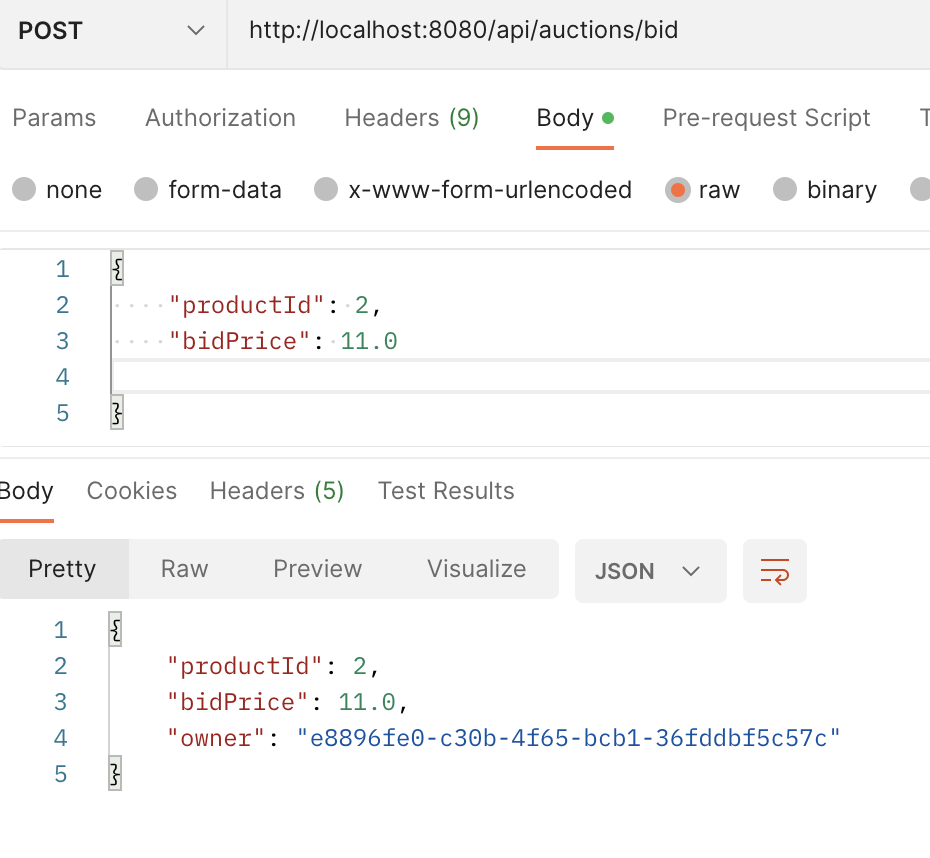
Sample: **token: <VALUE>** must be in headers.

{

"productId": 2,

"bidPrice": 11.0

}



* 1. URL for stopping Auction: <http://localhost:8080/api/auctions/stopAuction>

Sample:

{

"productId": 2

}

