Contents

Inventory Management System API Documentation

Overview

- Create and Read user roles.

  These roles need to be created manually in the database. Creating a "STAFF" role is mandatory as it is the role that is assigned to a User by default during the account creation. The "OWNER" role also needs to be created manually and assigned to a created user as endpoints for ROLE features can only be accessed by a USER with "OWNER" role. Then rest of the roles can be created through the API.

- Create (sign up) a user account and login.

  To create a user account, username, password, first name and last name should be provided. A new user account cannot be created with an existing username. The username must be at least 4 characters long and the password must be at least 6 characters long. The username and password will be used at login to receive a JWT token which will last for 30 minutes, that will be used for authentication in further APIs.

- Read and Update your own details.

  After logging into an account, you will be able to read your details and update your first and last name.

- Read all created users in the system, delete user, delete vendor and delete product.

  A user with "OWNER" role can read details of all the existing accounts in the system, delete a user, a vendor or a product.

  The deleted user, vendor or product are not removed from the database but are made unavailable for further usage. To signify that they have been removed, their records will have "_deleted" at the end of the username for the user and name for the product and vendor. While everything other than their id will be changed to null.

- Create, Read all or single vendor, Update vendor, Import records from csv to database, export from database to csv file.

To create a vendor, their name and contact must be provided. The contact must not belong to any existing vendor. The contact must be exactly 10 digits long and only contain numbers. Their relation to a product is created when a purchase order is made. The information about all the vendors or the vendor with provided id can be read.

Contact information and name of the vendor can be updated if valid contact information is provided that does not belong to any other vendor but themselves.

The records can be imported from a csv file containing only name and contact information. The file must not contain existing contact information.

The records from the database can be exported to a csv file.

- Create, Read all or single product, Update product, Import product records and vendor_product records from csv file to database and Export existing records from database to csv files.

  To create a product, product name and description must be provided. The product name must not belong to an existing product. The relation between a product and a vendor is made after placing an order.

  The information about all the products or the product with the provided id can be read.

  The name and description of the product of the product can be updated. The updated name must be new or only belong to themselves.

  Product records can be imported from a csv file containing name and description. The name must not belong to any existing product.

  Existing records from the products table can be exported to a csv file.

  The table containing relation between vendors and products can be exported and imported. The records to be imported must contain records with only existing vendor and product ids.

- Create, Read all or single order, update order status, and export from database to a csv file.

  To create an order, product id, vendor id, order quantity, unit price and order status must be provided. The product id and vendor id must belong to an available(nondeleted) and existing product and vendor. The order quantity and unit price cannot be 0. The order status can only be pending or delivered.

When an order is placed, a corresponding record in the vendor_product (table showing relation between product and vendor) table is affected. A record in the vendor_product table is considered unique if the combination of vendor id, product id and unit price is unique. If the placed order finds a record matching its vendor id, product id and unit price, the changes from the order will be made on that record on the vendor_product table, else a new record is created on that table.

If the order status is pending, the order quantity is not added to the stock quantity. If the order status is delivered, then the order quantity will be added to the stock quantity.

The update order status API also lets the order status to be changed from pending to delivered, which will update the stock quantity for the corresponding record in the vendor_product table. The update status can only be updated for the orders that are pending.

The details of all the orders or order with provided id can be read.

All the order records can be exported to a csv file.

Custom 403 Forbidden and 401 Unauthorized Responses

Error response 401 Unauthorized

```
{
   "401 Unauthorized": "Please login to access this resource."
}
```

You will be responded with this error message if you try to hit any endpoints other than signup or login without using JWT Token.

Error response 403 Forbidden

```
{
   "403 Forbidden": "Your provided role doesn't have access to this Resource."
}
```

You will be responded with this error message if a user with STAFF role tries to hit endpoints only available to OWNER role.

1. Role

Role endpoints are accessible only by OWNER role.

1.1.Get All Roles

Endpoint: GET localhost:8081/role/

Retrieve and display all the available roles in the system.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

```
[
   {
      "rid": 1,
      "name": "OWNER"
   },
   {
      "rid": 2,
      "name": "STAFF"
   }
]
```

1.2.Create A Role

Endpoint: POST localhost:8081/role/

Create a new role and save it in the system.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Request Body:

```
{
   "name" : "OWNER"
}
```

Success Response 201 Created

Role Created.

Creating a role with existing name sends out an error response.

Error Response 400 Bad Request

Role Name Already Exists.

2. Auth

2.1. Signup

Endpoint: localhost:8081/signup/

Create a user account.

Request Body:

```
{
   "username" : "testUser",
   "password" : "testUser",
   "firstName" : "testUser",
   "lastName" : "testUser"
}
```

Success response 201 Created

User Created. Login to view your details.

Creating a new user with existing username sends out a user already exists error response.

Request Body

```
{
   "username" : "testUser",
   "password" : "testUser",
   "firstName" : "testUser",
   "lastName" : "testUser"
}
```

Error Response 400 Bad Request

User Already Exists.

Username must have at least 4 characters and sends out an error response if it doesn't.

Request Body

```
{
   "username" : "ayu",
   "password" : "ayuzz",
   "firstName" : "aayush",
   "lastName" : "katwal"
}
```

Error Response: 400 Bad Request

Username Must be at least 4 Characters Long.

Password must be at least 6 characters long and sends out an error response if it isn't.

Request Body

```
{
   "username" : "ayuzz",
   "password" : "ayuzz",
   "firstName" : "aayush",
   "lastName" : "katwal"
}
```

Error Response 400 Bad Request

Password Must be at least 6 Characters Long.

2.2.Login

Endpoint: localhost:8081/login/

Log into an existing account using username and password.

This will provide you with a JWT token that will be used for authentication in further requests.

Request Body:

```
{
   "username" : "ayuzz1",
   "password" : "ayuzz1"
}
```

Succes Response 200 OK

```
eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJheXV6ejEiLCJpYXQiOjE3
MzIxNTg4NjcsImV4cCI6MTczMjE2MDY2N30.RpNy3e5mJDGRW_MPINnccU-
mQSdlioGP_OiTqKd5OxZZJNnu2Wz-5jiXusv7_AFfKvdpcvWJXiNRJySkgEL-Ow
```

If the provided username doesn't exist or username and password doesn’t match, it sends out an error response.

Response Body

```
{
   "username" : "ayuzz1",
   "password" : "ayuzz"
}
```

Error Response 400 Bad Request

Incorrect username or password.

3. User

3.1. Get Logged in User’s Details

Endpoint: GET localhost:8081/user/

Retrieve and display details of logged in User.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

```
{
   "uid": 2,
   "username": "ayuzz2",
   "firstName": "aayush",
   "lastName": "katwal",
   "roles": [
      {
         "rid": 2,
         "name": "STAFF"
      }
   ]
}
```

3.2.Update Logged in User’s Details

Update details about the logged in user. UserId, username and password can't be updated.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Request Body:

```
{
   "firstName" : "ayuzz2updated",
   "lastName" : "ayuzz2updated"
}
```

Success Response 200 OK

User Updated.

4. Admin

Admin endpoints are accessible only by OWNER role.

4.1.Get All Users

Endpoint: GET localhost:8081/admin/

Retrive and display products, orders and users list.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

```
[
   {
      "uid": 1,
      "username": "ayuzz1",
      "firstName": "aayush",
      "lastName": "katwal",
      "roles": [
         {
            "rid": 2,
            "name": "STAFF"
         },
         {
            "rid": 1,
            "name": "OWNER"
         }
      ]
   },
   {
      "uid": 2,
      "username": "ayuzz2",
      "firstName": "ayuzz2updated",
      "lastName": "ayuzz2updated",
        "roles": [
            {
                "rid": 2,
                "name": "STAFF"
            }
        ]
    }
]
```

Also Displays deleted users.

Success Response 200 OK

```
[
    {
        "uid": 1,
        "username": "ayuzz1",
        "firstName": "aayush",
        "lastName": "katwal",
        "roles": [
            {
                "rid": 2,
                "name": "STAFF"
            },
            {
                "rid": 1,
                "name": "OWNER"
            }
        ]
    },
    {
        "uid": 2,
        "username": "ayuzz2",
        "firstName": "ayuzz2updated",
        "lastName": "ayuzz2updated",
        "roles": [
            {
                "rid": 2,
                "name": "STAFF"
            }
        ]
    },
    {
        "uid": 3,
        "username": "testUser_deleted",
        "firstName": null,
        "lastName": null,
        "roles": [
            {
                "rid": 2,
                "name": "STAFF"
            }
        ]
    }
]
```

4.2.Delete A User

Endpoint: DELETE localhost:8081/admin/delete/user/{username}

Delete a user account with the provided username.

Deleting a user doesn't remove it from the Database, it deletes all the fields from its record and adds "_deleted" to the end of their username.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

User Deleted.

If nonexistent username is provided, User Not Found error response is sent out.

Error Response 400 Bad Request

User Not Found.

4.3.Delete A Product

Endpoint: DELETE localhost:8081/admin/delete/product/{productId}

Delete an existing product.

Deleting a product doesn't remove it from the database, it adds "_deleted" to its product name and the product is not allowed to be ordered further.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

Product Deleted.

Providing a nonexistent id sends out a Product not found error response.

Error Response 400 Bad Request

Product Not Found.

4.4.Delete A Vendor

Endpoint: DELETE localhost:8081/admin/delete/vendor/{vendorId}

Delete the vendor with the provided id.

Deleting a vendor doesn't remove it from the database, it removes their contact and adds "_deleted" to the end of their vendor's name. The deleted vendor cannot be used to purchase from any further.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

Vendor Deleted.

Providing nonexistent vendor id sends out Vendor not found error response.

Error Response 400 Bad Request

Vendor Not Found.

},

5. Vendor

5.1. Get All Vendors

Endpoint: GET localhost:8081/vendor/

Retrieve and display all available vendors and their details. It also provides the records of sold item related to the vendors.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

```
[
  {
    "vid": 2,
    "name": "updatedVendor",
    "contact": "1234567890",
    "vendorToProductResponseDTOList": [
      {
        "vpId": 1,
        "stockQuantity": 6,
        "unitPrice": 150,
        "productId": 1,
        "productName": "edited test product 1"
      },
      {
        "vpId": 3,
        "stockQuantity": 20,
        "unitPrice": 165,
        "productId": 1,
        "productName": "edited test product 1"
      }
    ]
  },
    {
        "vid": 3,
        "name": "vendor1",
        "contact": "9811111112",
        "vendorToProductResponseDTOList": [
            {
                "vpId": 2,
                "stockQuantity": 22,
                "unitPrice": 500,
                "productId": 3,
                "productName": "test product 3"
            }
        ]
    },
    {
        "vid": 4,
        "name": "vendor99",
        "contact": "9899999999",
        "vendorToProductResponseDTOList": []
    }
]
```

It also displays all the deleted vendors.

```
[
    {
        "vid": 2,
        "name": "updatedVendor",
        "contact": "1234567890",
        "vendorToProductResponseDTOList": [
            {
                "vpId": 1,
            "stockQuantity": 6,
            "unitPrice": 150,
            "productId": 1,
            "productName": "edited test product 1"
        },
        {
            "vpId": 3,
            "stockQuantity": 20,
            "unitPrice": 165,
            "productId": 1,
            "productName": "edited test product 1"
        }
    ]
},
{
    "vid": 3,
    "name": "vendor1",
    "contact": "9811111112",
    "vendorToProductResponseDTOList": [
        {
            "vpId": 2,
            "stockQuantity": 22,
            "unitPrice": 500,
            "productId": 3,
            "productName": "test product 3"
        }
    ]
},
{
    "vid": 4,
    "name": "vendor99_deleted_deleted",
        "contact": null,
        "vendorToProductResponseDTOList": []
    }
]
```

5.2. Get Vendor with ID

Endpoint: GET localhost:8081/vendor/{vendorId}

Retrieve and display the vendor having the provided id and their details.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

```
{
  "vid": 2,
  "name": "updatedVendor",
  "contact": "1234567890",
  "vendorToProductResponseDTOList": [
    {
      "vpId": 1,
      "stockQuantity": 6,
      "unitPrice": 150,
      "productId": 1,
      "productName": "edited test product 1"
    },
    {
      "vpId": 3,
      "stockQuantity": 0,
      "unitPrice": 165,
      "productId": 1,
      "productName": "edited test product 1"
    }
  ]
}
```

Nonexistent vendor id sends out a Vendor not found error message.

Error Response 400 Bad Request

Vendor Not Found.

5.3.Create Vendor

Endpoint: CREATE localhost:8081/vendor/

Create a vendor.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Request Body

```
{
   "name" : "vendor1",
   "contact" : "9811111111"
}
```

Success Response 201 Created

Vendor Created.

Creating another vendor with same name but different contact

Request Body

```
{
   "name" : "vendor1",
   "contact" : "  9811111112"
}
```

Success Response 201 Created

Vendor Created.

Contact number should be 10 digits long exactly. Else, it sends an error response

Request Body:

```
{
   "name" : "vendor1",
   "contact" : "981111111"
}
```

Error Response 400 Bad Request

Contact must contain 10 numbers.

Contact number should only contain digits. Else, it sends out an error response.

Request Body

```
{
   "name" : "vendor1",
   "contact" : "981111111a"
}
```

Error Response 400 Bad Request

Contact Must only contain numbers. For input string: "12345678s9"

Creating a duplicate vendor having existing contact number sends out an error response.

Request Body

```
{
   "name" : "vendor1",
   "contact" : "  9811111111"
}
```

Error Response 400 Bad Request

Vendor Already Exists.

5.4.Update Vendor Details

Endpoint: PUT localhost:8081/vendor/{vendorId}

Update details of the vendor with the provided id.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Request Body:

```
{
   "name" : "updatedVendor",
   "contact" : "1234567890"
}
```

Success Response 200 OK

Vendor Details Updated.

Providing a nonexistent vendor id sends out an error response.

Request Body:

```
{
   "name" : "updatedVendor",
   "contact" : "1234567890"
}
```

Error Response 200 OK

Vendor Not Found.

Providing contact information that is new or does not belong to that vendor will send out an error response.

Request Body:

```
{
   "name" : "updatedVendor",
   "contact" : "1234567890"
}
```

Error Response 200 OK

Provided contact already belongs to another vendor.

Contact number should be 10 digits long exactly. Else, it sends an error response

Request Body:

```
{
   "name" : "  updatedVendor",
   "contact" : "  123456789"
}
```

Error Response 400 Bad Request

Contact must contain 10 numbers.

Contact number should only contain digits. Else, it sends out an error response.

Request Body

```
{
   "name" : " updatedVendor ",
   "contact" : "  12345678s9"
}
```

Error Response 400 Bad Request

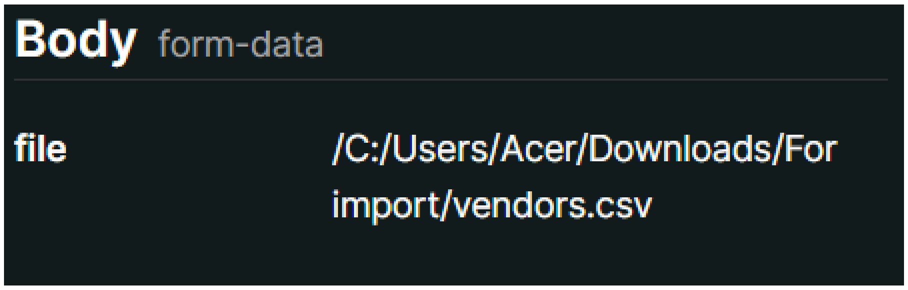
Contact Must only contain numbers. For input string: "12345678s9"

5.5.Import Vendors from CSV

Endpoint: POST localhost:8081/vendor/import

Import from a csv and persist the data in the database.

Request Body:



Body form-data

file	/C:/Users/Acer/Downloads/For import/vendors.csv

Success Response 200 OK

Products imported successfully.

Duplicate import sends out an error response.

Error importing products: Error processing CSV data: could not execute statement [Duplicate entry '9812121218' for key 'UKobo63jkxowc380j9srsmfa3dq'] [insert into vendors (contact,name) values (?,?)]; SQL [insert into vendors (contact,name) values (?,?)]; constraint [UKobo63jkxowc380j9srsmfa3dq]

5.6.Export Vendors to CSV

Endpoint: GET localhost:8081/vendor/export

Export id, name and contact of all the vendor from the database to vendors.csv file.

Success Response 200 OK

```
"Id","Name","Contact"
"2","updatedVendor3","1234567890"
"3","vendor1","9811111112"
"4","vendor99_deleted_deleted",
```

6. Product

6.1.Get All Products

Endpoint: GET localhost:8081/product/

Retrieve and display all available products. Also displays their relationship with vendors.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

```
[
   {
      "pid": 1,
      "name": "edited test product 1",
      "description": "edited description",
      "productToVendorResponseDTOList": [
         {
            "vpId": 1,
            "stockQuantity": 6,
            "unitPrice": 150,
            "vendorId": 2,
            "vendorName": "updatedVendor"
         },
         {
            "vpId": 3,
            "stockQuantity": 20,
            "unitPrice": 165,
            "vendorId": 2,
            "vendorName": "updatedVendor"
         }
      ]
   },
   {
        "pid": 2,
        "name": "test product 2",
        "description": "this is test product 2.",
        "productToVendorResponseDTOList": []
    },
    {
        "pid": 3,
        "name": "test product 3",
        "description": "this is test product 3.",
        "productToVendorResponseDTOList": [
            {
                "vpId": 2,
                "stockQuantity": 22,
                "unitPrice": 500,
                "vendorId": 3,
                "vendorName": "vendor1"
            }
        ]
    }
]
```

Also, deleted products are listed out as well.

Success Response 200 OK

```
[
    {
        "pid": 1,
        "name": "edited test product 1",
        "description": "edited description",
        "productToVendorResponseDTOList": [
            {
                "vpId": 1,
            "stockQuantity": 6,
            "unitPrice": 150,
            "vendorId": 2,
            "vendorName": "updatedVendor"
        },
        {
            "vpId": 3,
            "stockQuantity": 20,
            "unitPrice": 165,
            "vendorId": 2,
            "vendorName": "updatedVendor"
        }
    ]
},
{
    "pid": 2,
    "name": "test product 2_deleted",
    "description": "this is test product 2.",
    "productToVendorResponseDTOList": []
},
{
    "pid": 3,
    "name": "test product 3",
    "description": "this is test product 3.",
    "productToVendorResponseDTOList": [
        {
            "vpId": 2,
            "stockQuantity": 22,
            "unitPrice": 500,
            "vendorId": 3,
            "vendorName": "vendor1"
            }
          ]
      }
  ]
```

6.2. Get Product with ID

Endpoint: GET localhost:8081/product/{productId}

Retrieve and display details of product with given id and their corresponding vendor relationships.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response 200 OK

```
{
  "pid": 1,
  "name": "edited test product 1",
  "description": "edited description",
  "productToVendorResponseDTOList": [
    {
      "vpId": 1,
      "stockQuantity": 6,
      "unitPrice": 150,
      "vendorId": 2,
      "vendorName": "updatedVendor"
    },
    {
      "vpId": 3,
      "stockQuantity": 0,
      "unitPrice": 165,
      "vendorId": 2,
      "vendorName": "updatedVendor"
    }
  ]
}
```

Product Not Found error response for when nonexistent product id is passed.

Error Response 400 Bad Request

Product Not Found.

6.3. Create Product

Endpoint: POST localhost:8081/product/

Create a product.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Request Body:

```
{
   "name" : "test product 1",
   "description" : "this is test product 1."
}
```

Success Response 201 Created

Product Created.

Creating a product with an existing name sends out an error response.

Request Body:

```
{
   "name" : "test product 1",
   "description" : "this is test product 1."
}
```

Error Response 400 Bad Request

Product Already Exists.

6.4.Update Product

Endpoint: PUT localhost:8081/product/1

Update product details.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Request Body

```
{
   "name" : "edited test product 1",
   "description" : "edited description"
}
```

Succes Response 201 Created

Product Details Updated.

Nonexistent product id sends out an error response.

Request Body

```
{
   "name" : "edited test product 1",
   "description" : "edited description"
}
```

Error Response 400 Bad Request

Product Not Found.

The updated name must not belong to another product, or it will send out an error response.

Request Body

```
{
   "name" : "test product 2",
   "description" : "edited description"
}
```

Error Response 400 Bad Request

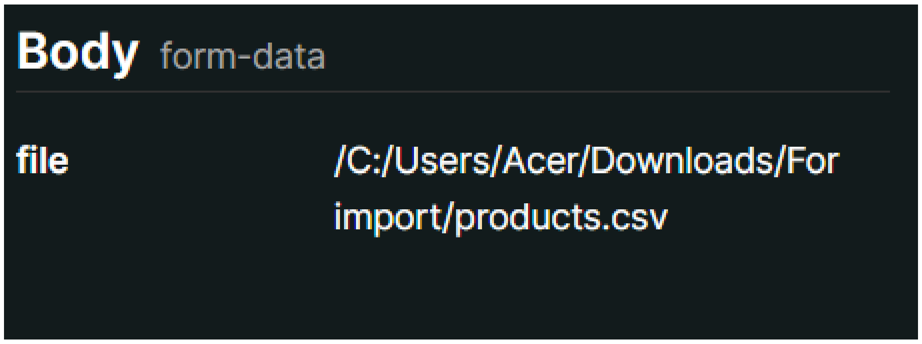
Another product with that name already exists. Try a different name.

6.5.Import Products from CSV

Endpoint: POST localhost:8081/product/import

Import products from csv file and persist the records in the database.

Request Body



Body form-data

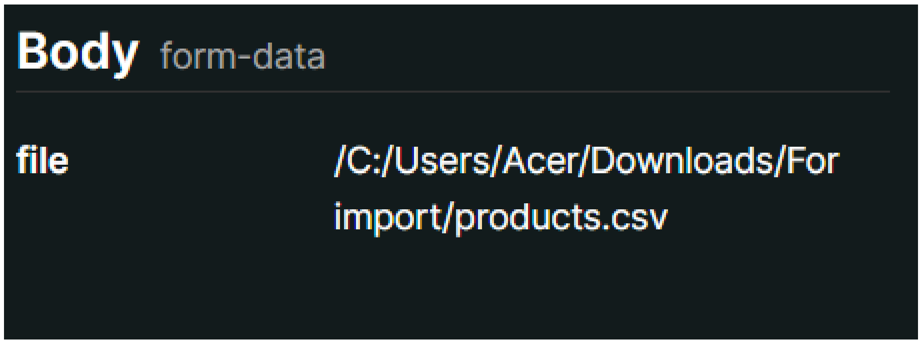
file	/C:/Users/Acer/Downloads/For import/products.csv

Success Response 200 OK

Products imported successfully.

Importing duplicate product sends an error response.

Request Body



Body form-data

file	/C:/Users/Acer/Downloads/For import/products.csv

Error Response 500 Internal Server Error

Error importing products: Error processing CSV data: could not execute statement [Duplicate entry 'imported product 1' for key 'UKo61fmio5yukmmiqgnxf8pnavn'] [insert into products (description,name) values (?,?)]; SQL [insert into products (description,name) values (?,?)]; constraint [UKo61fmio5yukmmiqgnxf8pnavn]

6.6.Export Products to CSV

Endpoint: GET localhost:8081/product/export

Export id, name and description of all the products from the database to products.csv file.

Success Response 200 OK

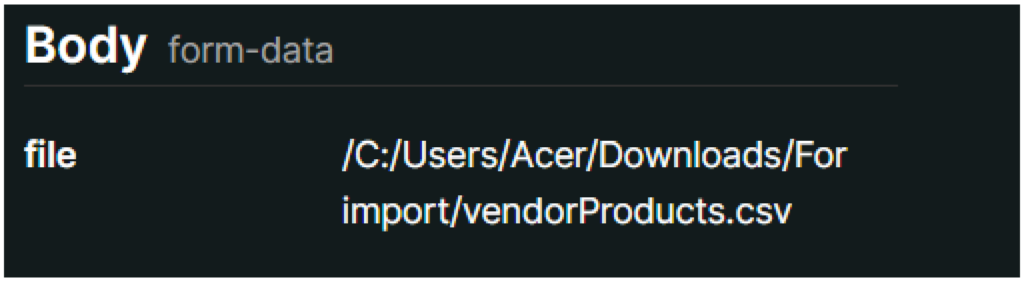
```
"Id","Name","Description"
"1","edited test product 1","edited description"
"2","test product 2_deleted","this is test product 2."
"3","test product 3","this is test product 3."
```

6.7.Import vendor_product table details from CSV

Endpoint: POST localhost:8081/product/vendor/import

Import records from csv file and persist in the vendor_product table in the database.

Request Body:



Body form-data

file	/C:/Users/Acer/Downloads/For import/vendorProducts.csv

Success Response

Products imported successfully.

6.8.Export vendor_product table details to CSV

Endpoint: GET localhost:8081/product/vendor/export

Export records from vendor_product table to vendorProducts.csv file.

Success Response 200 OK

```
"Id","StockQuantity","UnitPrice","Vendor ID","Product ID"
"1","6","150.0","1","2"
"2","22","500.0","3","3"
"3","20","165.0","1","2"
```

7. Order

7.1. Get All Orders

Endpoint: GET localhost:8081/order/

Read all orders that have been placed.

Success Response:

```
[
   {
      "oid": 1,
      "orderDate": "2024-11-20T13:06:39.000+00:00",
      "orderStatus": "delivered",
      "orderTotal": 750,
      "orderQuantity": 5,
      "unitPrice": 150,
      "productId": 1,
      "productName": "edited test product 1",
      "vendorId": 2,
      "vendorName": "updatedVendor"
   },
   {
      "oid": 2,
      "orderDate": "2024-11-20T13:07:44.000+00:00",
      "orderStatus": "delivered",
      "orderTotal": 150,
      "orderQuantity": 1,
      "unitPrice": 150,
      "productId": 1,
      "productName": "edited test product 1",
      "vendorId": 2,
      "vendorName": "updatedVendor"
   },
{
    "oid": 3,
    "orderDate": "2024-11-20T13:27:56.000+00:00",
    "orderStatus": "delivered",
    "orderTotal": 11000,
    "orderQuantity": 22,
    "unitPrice": 500,
    "productId": 3,
    "productName": "test product 3",
    "vendorId": 3,
    "vendorName": "vendor1"
},
{
    "oid": 4,
    "orderDate": "2024-11-20T14:27:01.000+00:00",
    "orderStatus": "delivered",
    "orderTotal": 1650,
    "orderQuantity": 10,
    "unitPrice": 165,
    "productId": 1,
    "productName": "edited test product 1",
    "vendorId": 2,
    "vendorName": "updatedVendor"
},
{
    "oid": 5,
    "orderDate": "2024-11-20T14:51:56.000+00:00",
    "orderStatus": "delivered",
    "orderTotal": 1650,
    "orderQuantity": 10,
    "unitPrice": 165,
        "productId": 1,
        "productName": "edited test product 1",
        "vendorId": 2,
        "vendorName": "updatedVendor"
    }
]
```

7.2.Get Purchase Order By ID

Endpoint: GET localhost:8081/order/{orderId}

Retrieve and display order details of the given order id.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Success Response

```
{
   "oid": 1,
   "orderDate": "2024-11-20T13:06:39.000+00:00",
   "orderStatus": "delivered",
   "orderTotal": 750,
   "orderQuantity": 5,
   "unitPrice": 150,
   "productId": 1,
   "productName": "edited test product 1",
   "vendorId": 2,
   "vendorName": "updatedVendor"
}
```

Nonexistent order id sends out an error response.

Error response

Order Not Found.

7.3.Create Purchase Order

Endpoint: POST localhost:8081/order/

Create an order.

When an order is created, if there is an existing record with the provided vendor id, product id and unit price in the vendor_product table, the existing record is used else a new record is created.

When order status is pending, ordered quantity will not be added to the stock quantity in its corresponding record in vendor_product table.

Bearer token required for authentication and authorization.

Use the JWT Token received while logging in.

Request Body

```
{
   "product" : {
      "pid" : 1
   },
   "vendor" : {
      "vid" : 2
   },
   "orderQuantity" : 1,
   "unitPrice" : 150,
   "orderStatus" : "delivered"
}
```

Success Response 201 Created

Order Created!

Only available product can be used else error message is provided.

```
{
   "product" : {
      "pid" : 10
   },
   "vendor" : {
      "vid" : 2
   },
   "orderQuantity" : 1,
   "unitPrice" : 150,
   "orderStatus" : "delivered"
}'
```

Error Response 400 Bad Request

The product doesn't exist or The product id belongs to a deleted product.

Only available vendor can be used else error message is provided.

```
{
    "product" : {
        "pid" : 1
    },
    "vendor" : {
        "vid" : 20
    },
    "orderQuantity" : 1,
    "unitPrice" : 150,
    "orderStatus" : "delivered"
}
```

Error Response 400 Bad Request

The Vendor doesn't exist, or The Vendor ID belongs to a Deleted Vendor.

Product id, vendor id, orderQuantity must be integers and unitPrice must be either integer or floating-point number. If not, an error response is sent out.

Request Body

```
{
    "product" : {
        "pid" : 3
    },
    "vendor" : {
        "vid" : 3
    },
    "orderQuantity" : "twenty two",
    "unitPrice" : 150,
    "orderStatus" : "pending"
}
```

Error Response 400 Bad Request

Invalid Input - Wrong data type provided.

Please check your response body.

Order status can only be pending or delivered, if not an error response is sent out.

Response Body

```
{
    "product" : {
        "pid" : 1
    },
    "vendor" : {
        "vid" : 2
    },
    "orderQuantity" : 0,
    "unitPrice" : 1,
    "orderStatus" : "test"
}
```

Error response 400 Bad Request

order status must be either "delivered" or "pending".

Unitprice and Order Quantity cannot be 0.

Request Body

```
{
    "product" : {
        "pid" : 1
    },
    "vendor" : {
        "vid" : 2
    },
    "orderQuantity" : 0,
    "unitPrice" : 1,
    "orderStatus" : "pending"
}
```
Error response 400 Bad Request

Unit price and order quantity can't be 0.

7.4.Update Order Status to Delivered

Endpoint: PUT localhost:8081/order/{orderId}

Update order status from pending to delivered.

This will add the order quantity to the stock quantity in its corresponding record in vendor_product table.

Response Body

```
{
   "updateStatusToDeliver": true
}
```

Success Response 201 Created

Order Delivered! Corresponding record in Vendor_Product Table has also been updated.

The provided id must be of existing order else an error response is sent out.

Error Response 400 Bad Request

Order Not Found.

If provided value of updateStatusToDeliver is false, error response is sent out.

Request Body

```
{
   "updateStatusToDeliver": false
}
```

Error Response 400 Bad Request

No need for changes while it’s not delivered.

If the order with the provided id has already been delivered, it also sends out an error response.

Error Response 400 Bad Request

Order has already been delivered.

7.5.Export Purchase Orders to CSV

Endpoint: localhost:8081/order/export

Export records from PurchaseOrders table to purchaseOrders.csv file.

Success Response 200 OK

```
"Id","OrderDate","OrderTotal","OrderQuantity","OrderStatus","UnitPrice","Product
ID","Vendor ID"
"1","2024-11-20 18:51:39.0","750.0","5","delivered","150.0","1","2"
"2","2024-11-20 18:52:44.0","150.0","1","delivered","150.0","1","2"
"3","2024-11-20 19:12:56.0","11000.0","22","delivered","500.0","3","3"
"4","2024-11-20 20:12:01.0","1650.0","10","delivered","165.0","1","2"
"5","2024-11-20 20:36:56.0","1650.0","10","delivered","165.0","1","2"
"6","2024-11-20 23:15:53.0","8250.0","50","pending","165.0","1","2"
"7","2024-11-21 18:55:27.0","8250.0","50","pending","165.0","1","2"
"8","2024-11-21 18:56:49.0","8250.0","50","pending","165.0","1","2"
```