API Documentation

1. Authentication API :

Endpoints:

POST /api/v1/users/signup :

Description: Creates a new user account.

Request Body:

```
{
  "name": "John Doe",
  "email": "john@example.com",
  "password": "password123"
}
```

Response:

```
{
   "status": "success",
   "token": "JWT_TOKEN",
   "data": {
   "user": {
    "name": "John Doe",
    "email": "john@example.com"
    }
  }
}
```

POST /api/v1/users/login:

Description: Logs in an existing user.

Request Body:

```
 {
  "email": "john@example.com",
  "password": "password123"
}
```

Response:

```
{
  "status": "success",
  "token": "JWT_TOKEN",
  "data": {
   "user": {
    "name": "John Doe",
    "email": "john@example.com"
    }
  }
}
```

POST /api/v1/users/upload_product_picture:

Description: Add user profile picture.

Request Body:

Form-data: profileImage

Protected Routes :

Middleware:

protect: Middleware to protect routes by verifying JWT token.

restrictTo: Middleware to restrict access based on user roles.

Usage Guidelines:

Signup:

Users should provide a name, email, and password in the request body.

Upon successful signup, a JWT token is generated and sent in the response along with user data.

Login:

Users should provide their email and password in the request body.

Upon successful login, a JWT token is generated and sent in the response along with user data.

Protect Middleware:

Protects routes by verifying the JWT token in the request headers.(Bearer Token)

Grants access to the protected route if the token is valid and the user exists.

RestrictTo Middleware:

Restricts access to routes based on user roles specified as arguments.

Only users with specified roles are granted access to the route.

2. Cart API :

Endpoints:

GET /api/v1/cart/

Description: Retrieves items in the user's cart.

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
  "status": "success",
  "data": [
    {
      "_id": "60983d60242cc9a42c7c823d",
      "product": {
        "_id": "60983035efb47391c3b11846",
        "name": "Product Name",
        "price": 100,
        "quantity": 5
      },
      "quantity": 2,
```

```
    "price": 200
   },
   ...
 ]
}
```


POST /api/v1/cart/

Description: Adds a product to the user's cart.

Request Headers:

Authorization: Bearer JWT_TOKEN

Request Body:


```
{
  "product_id": "60983035efb47391c3b11846",
}
```


Response:


```
{
  "status": "success",
  "numOfCartItems": 3,
  "data": {
    "_id": "60985a0f02e9b4e3a0b94b6a",
    "userId": "60983035efb47391c3b11846",
    "cartItems": [
      {
        "_id": "60985a0f02e9b4e3a0b94b6b",
        "product": "60983035efb47391c3b11846",
        "price": 100,
        "quantity": 1
      },
      ...
    ],
    "totalCartPrice": 300
  }
}
```

PUT /api/v1/cart/:itemId

Description: Update certain product quantity in cart.

Request Headers:

Authorization: Bearer JWT_TOKEN

Parameters: itemId

Request Body:

```
{
  "quantity": 5,
}
```

Response:

```
{
  "status": "success",
  "numOfCartItems": 3,
  "data": {
    "_id": "60985a0f02e9b4e3a0b94b6a",
    "userId": "60983035efb47391c3b11846",
    "cartItems": [
      {
        "_id": "60985a0f02e9b4e3a0b94b6b",
        "product": "60983035efb47391c3b11846",
        "price": 100,
        "quantity": 5
      },
      ...
    ],
    "totalCartPrice": 500
  }
}
```

DELETE /api/v1/cart/:itemId

Description: remove product from cart.

Request Headers:

Authorization: Bearer JWT_TOKEN

Parameters: cartId

DELETE /api/v1/cart/

Description: to delete whole cart

Request Headers:

Authorization: Bearer JWT_TOKEN

Usage Guidelines:

Get Items:

Send a GET request to **/api/v1/cart/** with the user's JWT token in the Authorization header to retrieve items in the user's cart.

The response contains an array of cart items with details such as product ID, name, price, quantity, etc.

Add Item:

Send a POST request to **/api/v1/cart/** with the product ID in request body and the user's JWT token in the Authorization header to add a product to the user's cart.

If the product is not found or out of stock, an error response is returned.

If the product is successfully added to the cart, the response contains the updated cart data including the total price and the number of cart items.

Update item quantity:

Send a PUT request to **/api/v1/cart/:itemId** with product Id as request parameter and the user's JWT token in authorization header to modify product quantity in user cart.

If cart not found an error response is returned.

If item not found an error response is returned.

Delete Item:

Send a DELETE request to **/api/v1/cart/:itemId** with product id as request parameter and the user's JWT token in authorization header to delete this product from the cart.

Delete Cart:

Send a DELETE request to **/api/v1/cart** with User's JWT token in authorization header to delete the cart.

3. Order API :

Endpoints:

POST /api/v1/order/

Description: Creates a new order.

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
  "status": "success",
  "data": {
    "_id": "60983d60242cc9a42c7c823d",
    "user": "60983035efb47391c3b11846",
    "cartItems": [
      {
        "product": "60983035efb47391c3b11846",
```

```
      "quantity": 1,
      "price": 200
     },
     ...
   ],
   "totalOrderPrice": 200
 }
}
```


GET /api/v1/order

Description: Retrieves all orders placed by the current user.

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
 "status": "success",
 "data": [
  {
   "_id": "60983d60242cc9a42c7c823d",
   "user": "60983035efb47391c3b11846",
   "cartItems": [
    {
     "product": "60983035efb47391c3b11846",
     "quantity": 1,
     "price": 200
    },
    ...
   ],
   "totalOrderPrice": 200
  },
  ...
 ]
}
```


GET /api/v1/order/delivered

Description: Retrieves all delivered orders placed by the current user.

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
 "status": "success",
 "data": [
  {
   "_id": "60983d60242cc9a42c7c823d",
   "user": "60983035efb47391c3b11846",
   "cartItems": [
```

```
      {
        "product": "60983035efb47391c3b11846",
        "quantity": 1,
        "price": 200
      },
      ...
    ],
    "totalOrderPrice": 200
  },
  ...
 ]
}
```


GET /api/v1/order/undelivered

Description: Retrieves all undelivered orders placed by the current user.

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
 "status": "success",
 "data": [
  {
   "_id": "60983d60242cc9a42c7c823d",
   "user": "60983035efb47391c3b11846",
   "cartItems": [
    {
     "product": "60983035efb47391c3b11846",
     "quantity": 1,
     "price": 200
    },
    ...
   ],
   "totalOrderPrice": 200
  },
  ...
 ]
}
```


GET /api/v1/order/:id

Description: Retrieves details of certain order of the user.

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
```

```
 "status": "success",
 "data": [
  {
   "_id": "60983d60242cc9a42c7c823d",
   "user": "60983035efb47391c3b11846",
   "cartItems": [
    {
     "product": "60983035efb47391c3b11846",
     "quantity": 1,
     "price": 200
    },
    ...
   ],
   "totalOrderPrice": 200
  },
  ...
 ]
}
```


GET /api/v1/order/pay/:order_id

Description: Update certain order status to paid.

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
 "status": "success",
 "data": [
  {
   "_id": "60983d60242cc9a42c7c823d",
   "user": "60983035efb47391c3b11846",
   "cartItems": [
    {
     "product": "60983035efb47391c3b11846",
     "quantity": 1,
     "price": 200
    },
    ...
   ],
   "totalOrderPrice": 200
  },
  ...
 ]
}
```

GET /api/v1/order/deliver/:order_id

Description: Update certain order status to delivered.

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
 "status": "success",
 "data": [
  {
   "_id": "60983d60242cc9a42c7c823d",
   "user": "60983035efb47391c3b11846",
   "cartItems": [
    {
     "product": "60983035efb47391c3b11846",
     "quantity": 1,
     "price": 200
    },
    ...
   ],
   "totalOrderPrice": 200
  },
  ...
 ]
}
```


GET /api/v1/order/dashboard

Description: returns details from certain start date to certain end date

Request Headers:

Authorization: Bearer JWT_TOKEN

Query parameters: startDate, endDate

Response:


```
{
 "status": "success",
 "data": {
dashboardData: {
        totalSales: 5,
        totalRevenue: 4000 ,
        totalCustomers: 2
        },
recentSales:   {
    "_id": "60983d60242cc9a42c7c823d",
    "user": "60983035efb47391c3b11846",
    "cartItems": [
     {
```

```
        "product": "60983035efb47391c3b11846",
        "quantity": 10,
        "price": 200
       },
       ...
     ],
     "totalOrderPrice": 2000
   },
   ...
        ,
topSellingProducts:   {

        "product": "60983035efb47391c3b11846",
        "quantity": 10,
        "price": 200
       },
   ...
        }
}
```


GET /api/v1/order/all-orders

Description: retrieve all orders details from all users

Request Headers:

Authorization: Bearer JWT_TOKEN

Response:


```
{
  "status": "success",
  "data": [
    {
      "_id": "60983d60242cc9a42c7c823d",
      "user": "60983035efb47391c3b11846",
      "cartItems": [
        {
          "product": "60983035efb47391c3b11846",
          "quantity": 1,
          "price": 200
        },
        ...
      ],
      "totalOrderPrice": 200
    },
    ...
  ]
}
```

Usage Guidelines:

Create Order:
Send a POST request to **/api/v1/order/create** with the user's JWT token in the Authorization header to create a new order.
The response contains the newly created order data including cart items and total order price.
View All Orders:
Send a GET request to **/api/v1/order** with the user's JWT token in the Authorization header to retrieve all orders placed by the current user.
View Delivered Orders:
Send a GET request to **/api/v1/order/delivered** with the user's JWT token in the Authorization header to retrieve all delivered orders placed by the current user.
View Undelivered Orders:
Send a GET request to **/api/v1/order/undelivered** with the user's JWT token in the Authorization header to retrieve all undelivered orders placed by the current user.
Update Order Delivery status:
Send a GET request to **/api/v1/order/deliver/:order_id** with the user's JWT token in the Authorization header to update certain order delivery status.
Update Order Payment status:
Send a GET request to **/api/v1/order/pay/:order_id** with the user's JWT token in the Authorization header to update certain order payment status.
Get Dashboard:
Send a GET request to **/api/v1/order/dashboard** with the user's JWT token in the Authorization header to retrieve dashboard of sales and revenues and most sold product.
View All Orders:
Send a GET request to **/api/v1/order/all-orders** with the user's JWT token in the Authorization header to retrieve all data of all orders.
View Certain Order:
Send a GET request to **/api/v1/order/:id** with the user's JWT token in the Authorization header to retrieve all details of a certain order of authenticated user.

4. Product API :

Endpoints:

GET /api/v1/products/

Description: Retrieves all products.

Request Query Parameters (Optional):

price[gte]: Filter products with a price greater than or equal to the specified value.

price[lte]: Filter products with a price less than or equal to the specified value.

sort: Sort products based on specified fields (e.g., price, createdAt).

fields: Select specific fields to include in the response.

page: Specify the page number for paginated results.

limit: Specify the maximum number of products per page.

Response:


```
{
  "status": "success",
  "data": [
    {
      "_id": "60983d60242cc9a42c7c823d",
      "name": "Product Name",
      "description": "Product Description",
      "price": 100,
      ...
    },
    ...
  ]
}
```


GET /api/v1/products/:id

Description: Retrieve specific product data by ID.

Request Parameters:

id: Product ID

Response:


```
{
  "status": "success",
  "data": {
    "_id": "60983d60242cc9a42c7c823d",
    "name": "Product Name",
    "description": "Product Description",
    "price": 100,
    ...
  }
}
```


POST /api/v1/products

Description: Adds new product.

Request Body :


```
{
```

```
  "name": "Product Name",
  "description": "Product Description",
  "price": 100,
  “gender”: female,
“age”: 2,
“breed”: “yes”,
“price”: 3923
}
```


Response:


```
{
  "status": "success",
  "data": {
    "_id": "60983d60242cc9a42c7c823d",
    "name": "Product Name",
    "description": "Product Description",
    "price": 100,
    ...
  }
}
```


POST /api/v1/products/upload_product_picture

Description: Add product picture.

Request Parameters:

id: Product ID

Request Body:

Form-data: productImage

PUT /api/v1/products/:id

Description: Updates an existing product price or description.

Request Parameters:

id: Product ID

Request Body:


```
{
  "description": "Updated Product Description",
  "price": 120,
}
```


Response:


```
{
  "status": "success",
  "data": {
    "_id": "60983d60242cc9a42c7c823d",
    "name": "Updated Product Name",
    "description": "Updated Product Description",
```

```
   "price": 120,
   ...
  }
}
```


DELETE/api/v1/products/:id

Description: Deletes an existing product from products list.
Request Parameters:
id: Product ID
Response:


```
{
  "status": "success",
  "data": null
}
```


Usage Guidelines:

Get All Products:
Send a GET request to **/api/v1/products** to retrieve all products.
Use query parameters to filter, sort, and paginate results.
Get Product by ID:
Send a GET request to **/api/v1/products/:id** to retrieve a specific product by its ID.
Add Product:
Send a POST request to **/api/v1/products** with the product details in the request body to add a new product.
Update Product:
Send a PATCH request to **/api/v1/products/:id** with the updated product details in the request body to modify an existing product.
Delete Product:
Send a DELETE request to **/api/v1/products/:id** to delete an existing product by its ID.