



VIT[®]

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

INTERNET AND WEB PROGRAMMING

(CSE3002)

Title: Organic Shopping

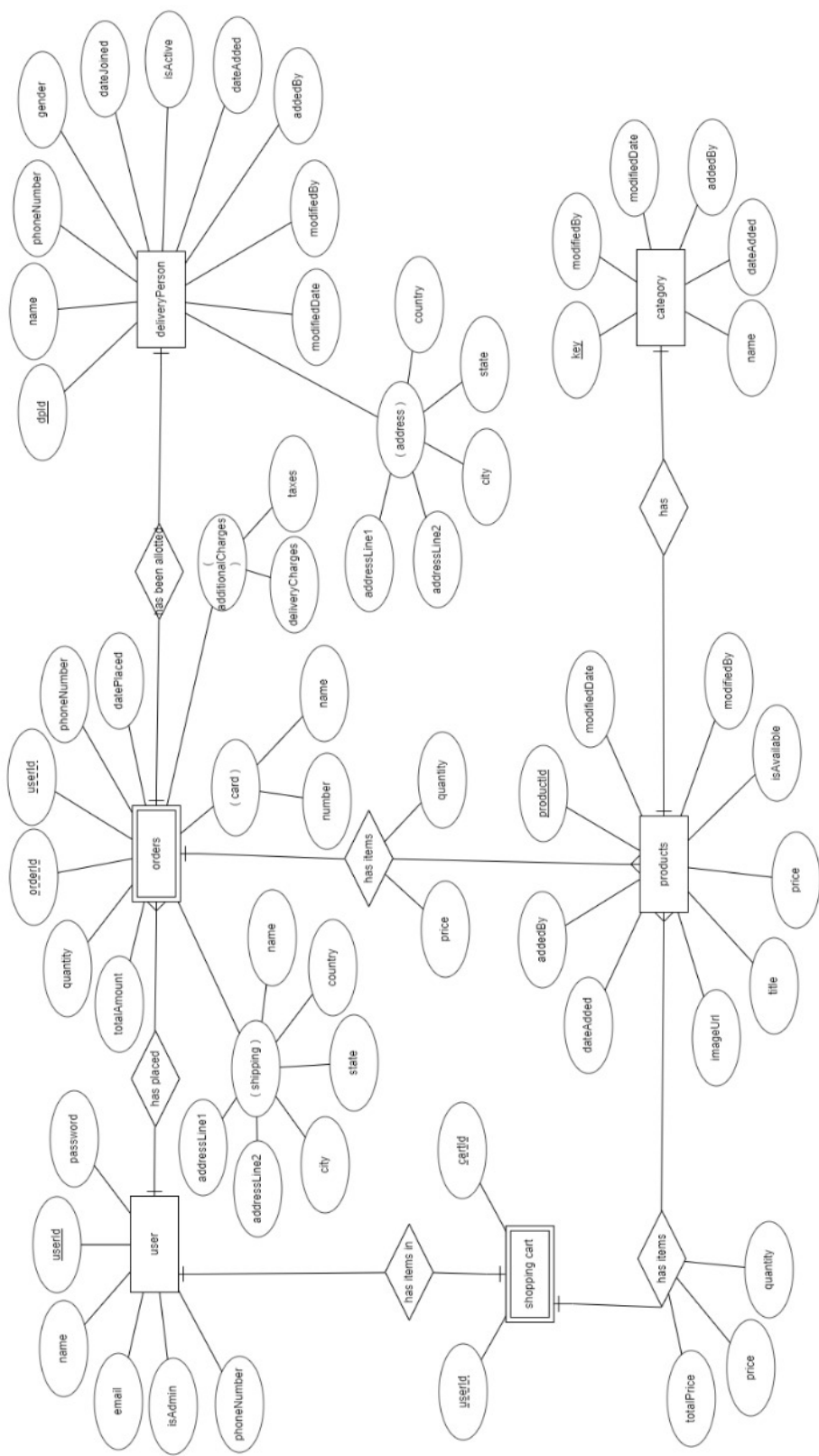
REVIEW-2

Slot: B1+TB1

Submitted By:
ROHAN MITTAL (18BCE0503)

Submitted To:
PROF. LYDIA JANE G

ER DIAGRAM



DB QUERIES

User

Query to Register User

```
router.post('/users/register', (req, res) => {
  res.header("Access-Control-Allow-Origin", "*");
  console.log(req.body)
  const users = new User(req.body)
  users.save().then(() => {
    res.status(201)
    res.send(users)
  }).catch((e) => {
    res.status(400)
    res.send(e)
  })
})
```

Query to Login User

```
router.post('/users/login', async (req, res) => {
  // res.header("Access-Control-Allow-Origin", "*");
  console.log(req.body)
  try{
    const user = await User.findByCredentials(req.body.email, req.body.password);
    const token = await user.generateAuthToken()
    res.send({user, token})
  }catch(e){
    res.status(400).send('Account not found')
  }
})
```

Get All User Data

```
router.get('/users', (req, res) => {

  res.header("Access-Control-Allow-Origin", "*");
  User.find({}).then((users) => {
    res.send(users)
  }).catch((e) => {
    res.status(500)
    res.send()
  })
})
```

Get Single User Data (called upon User Login to get Details)

```
router.get('/users/:id', (req, res) => {
  res.header("Access-Control-Allow-Origin", "*");
  const _id = req.params.id
```

```

    User.findById(_id).then((user) => {
      if(!user){
        return res.status(404).send()
      }
      res.status(200).send(user)
    }).catch((e) => {
      res.status(500).send()
    })
  })
})

```

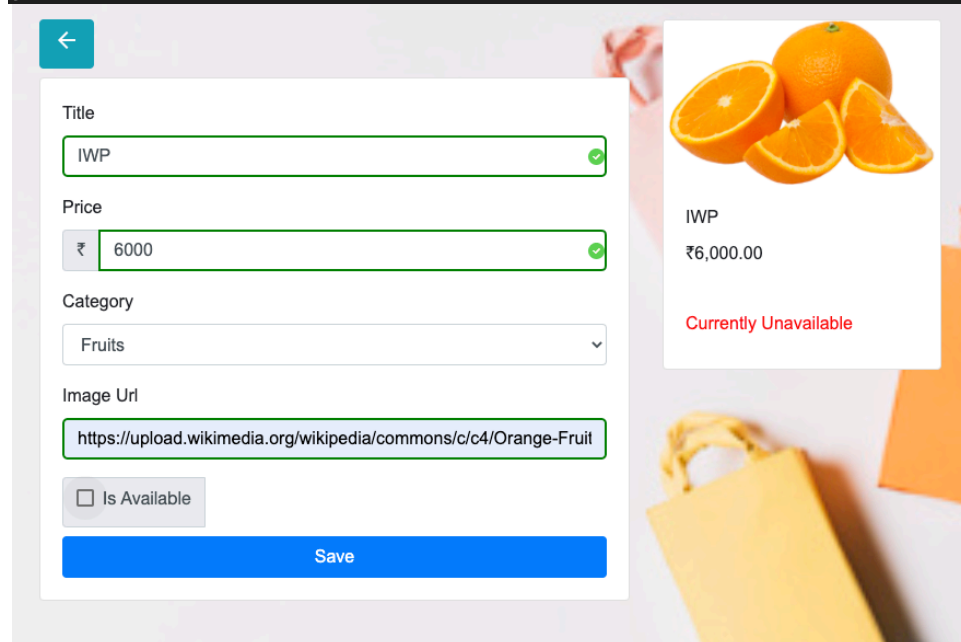
Products

Add/Post a product to database

```

router.post('/products', (req, res) => {
  res.header("Access-Control-Allow-Origin", "*");
  console.log(req.body)
  const products = new Product(req.body)
  products.save().then(() => {
    res.status(201)
    res.send(products)
  }).catch((e) => {
    res.status(400)
    res.send(e)
  })
})

```



The screenshot displays a web application interface for adding a product. On the left, there is a form with the following fields:

- Title:** A text input field containing "IWP" with a green checkmark icon on the right.
- Price:** A text input field containing "6000" with a green checkmark icon on the right. A small "₹" symbol is visible to the left of the input.
- Category:** A dropdown menu with "Fruits" selected.
- Image Url:** A text input field containing "https://upload.wikimedia.org/wikipedia/commons/c/c4/Orange-Fruit".
- Is Available:** A checkbox that is currently unchecked.
- Save:** A blue button at the bottom of the form.

On the right, there is a product card with a background image of oranges and a yellow shopping bag. The card contains the following information:

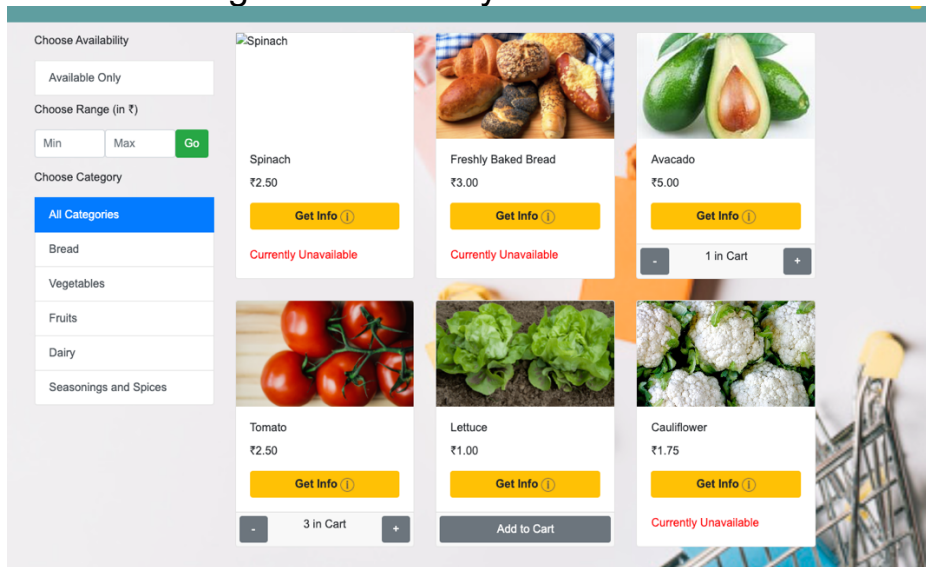
- Title:** IWP
- Price:** ₹6,000.00
- Status:** Currently Unavailable (text in red)

Get Available Products from database

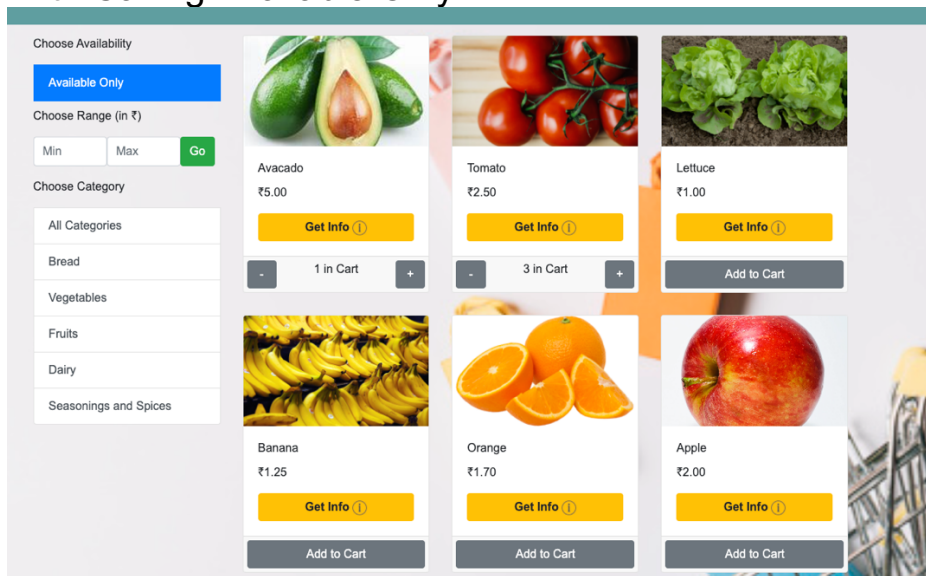
```
router.get('/products/available', (req,res) => {

    res.header("Access-Control-Allow-Origin", "*");
    Product.find({isAvailable: true}).then((products) => {
        res.send(products)
    }).catch((e) => {
        res.status(500)
        res.send()
    })
})
})
```

Without calling Available Only



















With Calling Available Only



Get All Products from database

```
router.get('/products/all', (req,res) => {  
  
  res.header("Access-Control-Allow-Origin", "*");  
  Product.find({}).then((products) => {  
    res.send(products)  
  }).catch((e) => {  
    res.status(500)  
    res.send()  
  })  
})  
})
```

Add <input type="text" value="Search"/>				
SNo	Title	Price	Available?	Edit
1	Spinach	2.5	<input type="checkbox"/>	 
2	Freshly Baked Bread	3	<input type="checkbox"/>	 
3	Avacado	1.75	<input checked="" type="checkbox"/>	 
4	Tomato	2.5	<input checked="" type="checkbox"/>	 
5	Lettuce	1	<input checked="" type="checkbox"/>	 
6	Cauliflower	1.75	<input type="checkbox"/>	 
7	Banana	1.25	<input checked="" type="checkbox"/>	 

SNo	Title	Price	Available?	Edit
21	IWP	6000	<input type="checkbox"/>	 

Items per page: 5 21 – 21 of 21 < < > >

Choose Availability

Available Only

Choose Range (in ₹)

Min Max Go

Choose Category

All Categories

Bread


Vegetables

Fruits

Dairy

Seasonings and Spices

Spinach




Spinach

₹2.50

Get Info ⓘ

Currently Unavailable




Freshly Baked Bread

₹3.00

Get Info ⓘ

Currently Unavailable




Avacado

₹5.00

Get Info ⓘ

1 in Cart




Tomato

₹2.50

Get Info ⓘ

3 in Cart




Lettuce

₹1.00

Get Info ⓘ

Add to Cart



Cauliflower

₹1.75

Get Info ⓘ

Currently Unavailable

Get Data of Product with a unique ID (used to get details of single product when editing details of product)

```
router.get('/products/:id', async (req,res) => {
  res.header("Access-Control-Allow-Origin", "*");
  const _id = req.params.id
  try{
    const product = await Product.findById(_id)
    if(!product){
      return res.status(404).send()
    }
    res.status(200).send(product)
  }catch(e){
    res.status(500).send(e)
  }
})
```

The screenshot shows a REST client interface with the following details:

- Method:** GET
- URL:** localhost:3000/products/5f1f06d9a3ca483a6a6ad9ed
- Send Button:** A blue button labeled "Send".
- Params Tab:** Selected, showing a table with columns KEY, VALUE, and DESCRIPTION.
- Body Tab:** Selected, showing a JSON response in "Pretty" format.
- Status:** 200 OK, Time: 11 ms, Size: 613 B.
- Save Button:** A red button labeled "Save".

The JSON response in the Body tab is as follows:

```
{
  "price": 1.75,
  "isAvailable": true,
  "_id": "5f1f06d9a3ca483a6a6ad9ed",
  "imageUrl": "https://tse2.mm.bing.net/th?id=0IP.iUlhfv5Iala0vFIQ-_W4kgHaCQ5pid=Api&P=06w=202&h=172",
  "category": "Fruits",
  "title": "Avacado",
  "dateAdded": "2020-07-27T16:54:49.000Z",
  "addedBy": "rohanmittal01@gmail.com",
  "modifiedDate": "2020-07-27T17:53:05.000Z",
  "modifiedBy": "rohanmittal01@gmail.com",
  "__v": 0
}
```

Update Data of Product with a unique ID

```
router.patch('/products/:id', async (req,res) => {
  try{
    const product = await Product.findByIdAndUpdate(req.params.id, req.body, {new:
true, runValidators: true})
    if(!product){
      return res.status(404).send()
    }
    res.send(product)
  }catch(e){
    res.status(400).send(e)
  }
})
```

Modified price of Avocado from 1.75 to 5

PATCH localhost:3000/products/5f1f06d9a3ca483a6a6ad9ed

Params Authorization Headers (8) **Body** Pre-request Script Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "price": 5,
3   "isAvailable": true,
4   "imageUrl": "https://tse2.mm.bing.net/th?id=0IP.iUlhfv5Iala0vFIQ-_W4kgHaGQ&pid=Api&P=06w=2026h=172",
5   "category": "Fruits",
6   "title": "Avacado",
7   "dateAdded": "2020-07-27T16:54:49.000Z",
8   "addedBy": "rohanmittal01@gmail.com",
9   "modifiedDate": "2020-07-27T17:53:05.000Z",
10  "modifiedBy": "rohanmittal01@gmail.com"
11 }
```

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "price": 5,
3   "isAvailable": true,
4   "_id": "5f1f06d9a3ca483a6a6ad9ed",
5   "imageUrl": "https://tse2.mm.bing.net/th?id=0IP.iUlhfv5Iala0vFIQ-_W4kgHaGQ&pid=Api&P=06w=2026h=172",
6   "category": "Fruits",
7   "title": "Avacado",
8   "dateAdded": "2020-07-27T16:54:49.000Z",
9   "addedBy": "rohanmittal01@gmail.com",
10  "modifiedDate": "2020-07-27T17:53:05.000Z",
11  "modifiedBy": "rohanmittal01@gmail.com",
12  "__v": 0
13 }
```



Delete Data of Product with a unique ID

```
router.delete('/products/:id', async (req, res) => {
  try{
    const product = await Product.findByIdAndDelete(req.params.id);
    console.log(product)
    if(!product){
      return res.status(404).send()
    }
    res.send(product)
  }catch{
    res.status(500).send()
  }
})
```

Before(total is 21 products)

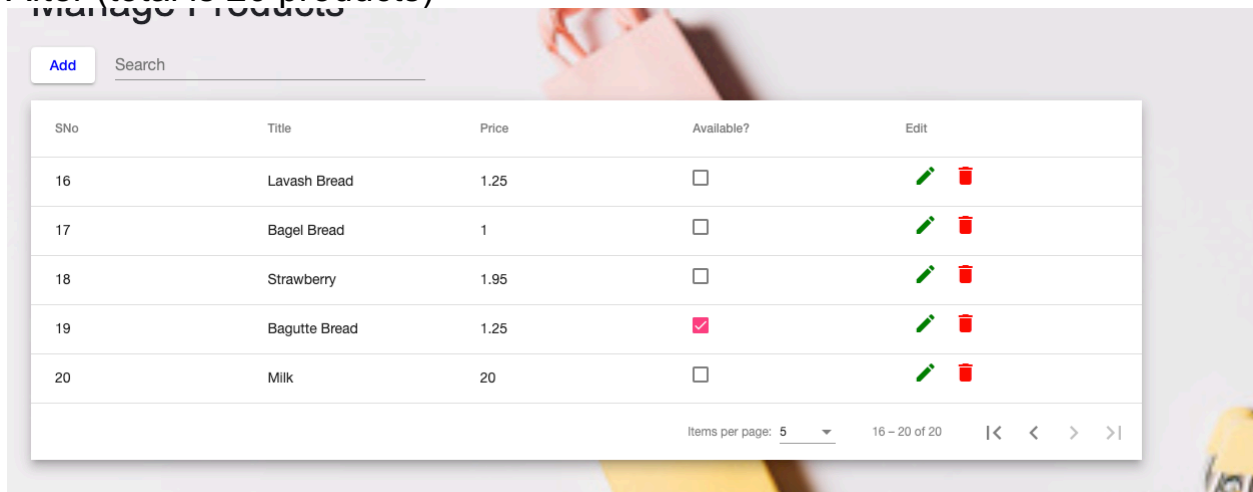
Manage Products











Add Search

SNo	Title	Price	Available?	Edit
21	IWP	6000	<input type="checkbox"/>	 

Items per page: 5 21 - 21 of 21 |< < > >|

After (total is 20 products)



SNo	Title	Price	Available?	Edit
16	Lavash Bread	1.25	<input type="checkbox"/>	 
17	Bagel Bread	1	<input type="checkbox"/>	 
18	Strawberry	1.95	<input type="checkbox"/>	 
19	Bagutte Bread	1.25	<input checked="" type="checkbox"/>	 
20	Milk	20	<input type="checkbox"/>	 

Items per page: 5 16 - 20 of 20 |< < > >|

Shopping Cart

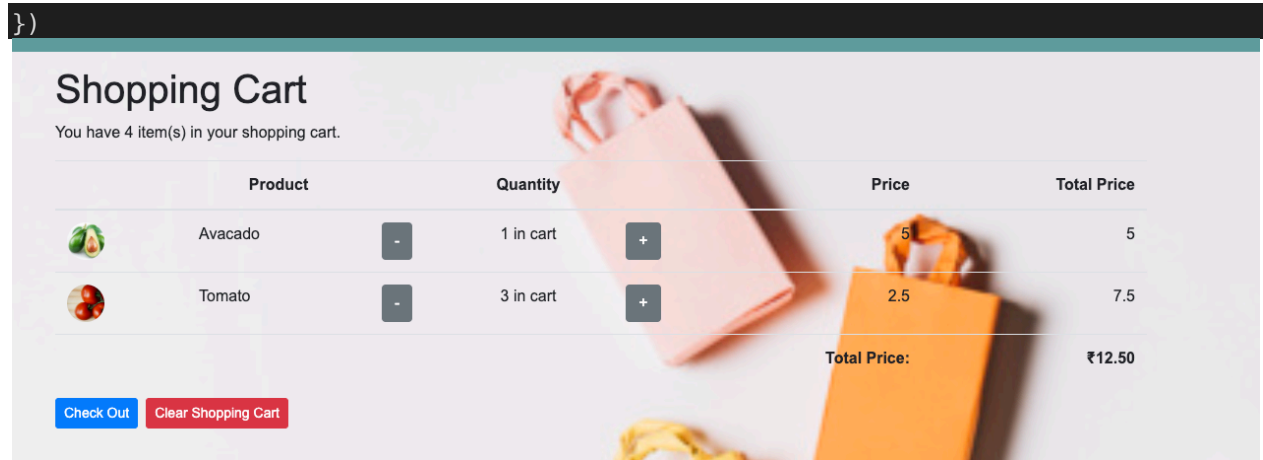
Create New Shopping Cart for user

```
router.post('/shopping-cart', (req, res) => {
  // res.header("Access-Control-Allow-Origin", "*");
  console.log(req.body)
  const cart = new Cart(req.body)
  cart.save().then(() => {
    res.status(201)
    res.send(cart)
  }).catch((e) => {
    res.status(400)
    res.send(e)
    console.log(e)
  })
})
```

Get Data of Shopping Cart with a unique User ID

```
router.get('/shopping-cart/:id', async (req, res) => {

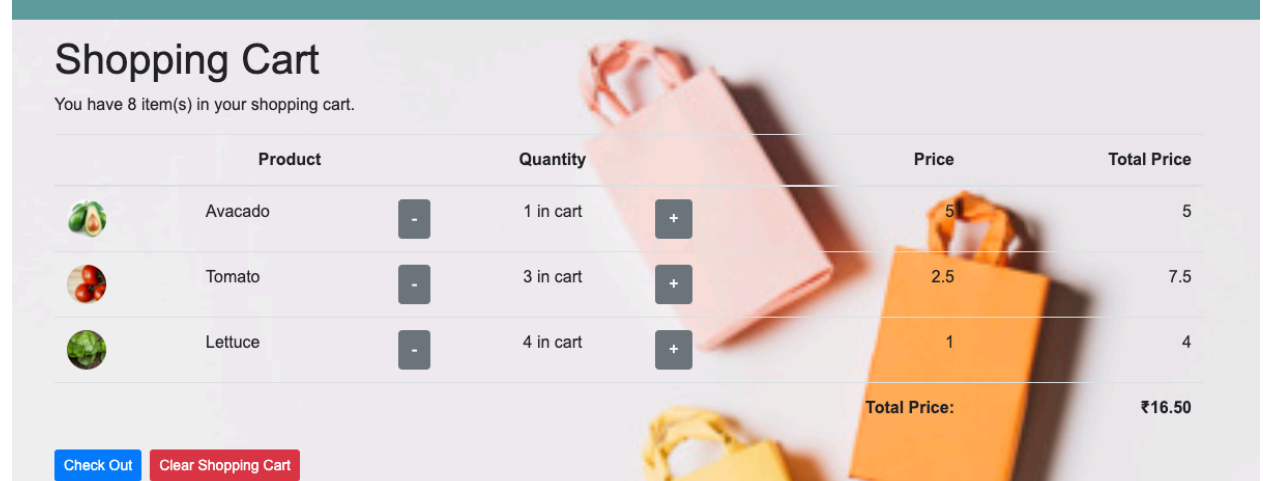
  res.header("Access-Control-Allow-Origin", "*");
  const _id = req.params.id
  try{
    const cart = await Cart.findOne({userId: _id})
    if(!cart){
      return res.status(404).send()
    }
    res.status(200).send(cart)
  }catch(e){
    res.status(500).send(e)
  }
})
```



Update Data of Shopping Cart with a unique User ID

```
router.patch('/shopping-cart/:id', async (req,res) => {
  try{
    const cart = await Cart.findOneAndUpdate({userId: req.params.id}, req.body,
{new: true, runValidators: true})
    if(!cart){
      return res.status(404).send()
    }
    res.send(cart)
  }catch(e){
    res.status(400).send(e)
  }
})
```

After adding lettuce to cart



Delete Data of Shopping Cart with a unique User ID

```
router.delete('/shopping-cart/:id', async (req, res) => {
  try{
    const cart = await Cart.findOneAndDelete({userId: req.params.id});
    console.log(cart)
    if(!cart){
      return res.status(404).send()
    }
    res.send(cart)
  }catch{
    res.status(500).send()
  }
})
```

Shopping Cart

Sorry!
No items in Cart.

Add Items to Cart

Orders

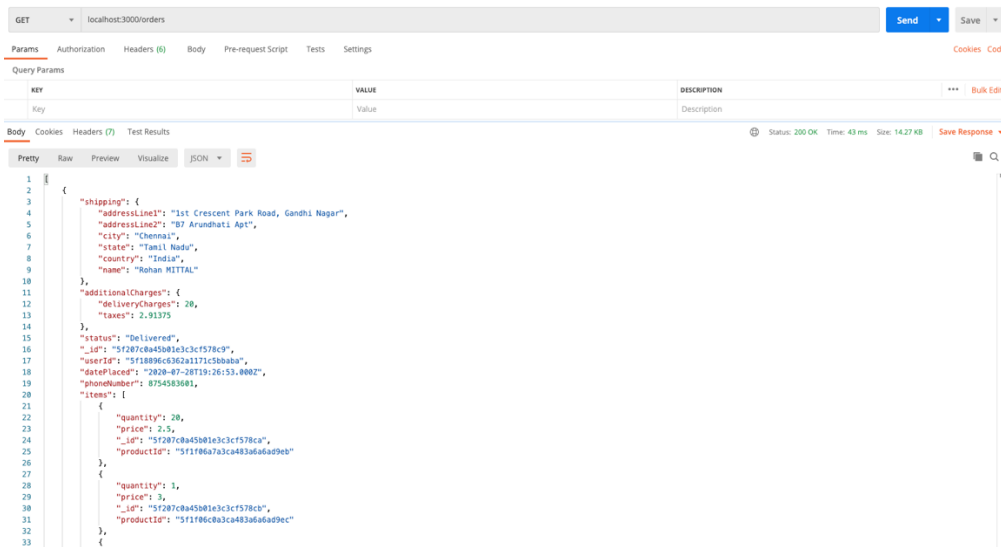
Add/post a new Order to database

```
router.post('/orders', (req, res) => {
  // res.header("Access-Control-Allow-Origin", "*");
  console.log(req.body)
  const orders = new Orders(req.body)
  orders.save().then(() => {
    res.status(201)
    res.send(orders)
  }).catch((e) => {
    res.status(400)
    res.send(e)
    console.log(e)
  })
})
```

Get All Orders Data

```
router.get('/orders', (req, res) => {

  res.header("Access-Control-Allow-Origin", "*");
  Orders.find({}).then((orders) => {
    res.send(orders)
  }).catch((e) => {
    res.status(500)
    res.send()
  })
})
```



Get Data of Orders with a unique User ID

```

router.get('/orders/:user', async (req,res) => {
  res.header("Access-Control-Allow-Origin", "*");
  const _id = req.params.user
  console.log(_id);
  try{
    const orders = await Orders.find({userId: _id})
    if(!orders){
      return res.status(404).send()
    }
    res.status(200).send(orders)
  }catch(e){
    res.status(500).send(e)
  }
})

```

Search					
SNo	Date	Quantity	Amount	Status	Edit
1	7/29/2020, 12:56:53 AM		78.28	Delivered	View more
2	7/29/2020, 12:59:39 AM		78.28	Order Placed	View more
3	7/29/2020, 1:01:18 AM		78.28	Order Placed	View more
4	7/29/2020, 1:18:16 AM		22.63	Order Placed	View more
5	7/30/2020, 1:53:14 AM		43.36	Order Placed	View more

Items per page: 5 1 - 5 of 12 < > >>

Get Data of Orders with a unique Order Id

```
router.get('/order/id/:id', (req,res) => {
  res.header('Access-Control-Allow-Origin', "*");
  const _id = req.params.id
  console.log(_id)
  Orders.findById(_id).then((order) => {
    console.log('hi')
    console.log(order)
    if(!order){
      return res.status(404).send()
    }
    res.status(200).send(order)
  }).catch((e) => {
    res.status(500).send()
  })
})
```

The screenshot shows a REST client interface with a GET request to `localhost:3000/orders/id/5f21db1882cc691e34a1b4d5`. The response is a JSON object with the following structure:

```
{
  "shipping": {
    "addressLine1": "1st Crescent Park Road, Gandhi Nagar",
    "addressLine2": "B7 Arundhati Apt",
    "city": "Chennai",
    "state": "Tamil Nadu",
    "country": "India",
    "name": "Rohan MITTAL"
  },
  "additionalCharges": {
    "deliveryCharges": 20,
    "taxes": 1.168125
  },
  "status": "Order Placed",
  "_id": "5f21db1882cc691e34a1b4d5",
  "userId": "5f18896c6362a1171c5bbaba",
  "datePlaced": "2020-07-29T20:23:14.000Z",
  "phoneNumber": 8754583601,
  "items": [
    {
      "quantity": 1,
      "price": 1.75,
      "_id": "5f21db1882cc691e34a1b4d6",
      "productId": "5f1f0729a3ca483a6a6ad9f0"
    },
    {
      "quantity": 1,
      "price": 1.25,
      "_id": "5f21db1882cc691e34a1b4d7",
      "productId": "5f1f0740a3ca483a6a6ad9f1"
    },
    {
      "quantity": 11,
      "price": 1.75,
      "_id": "5f21db1882cc691e34a1b4d8",
      "productId": "5f1f06d9a3ca483a6a6ad9ed"
    }
  ]
}
```

Update Data of Orders with a unique Order ID

```
router.patch('/orders/:id', async (req,res) => {
  try{
    const orders = await Orders.findByIdAndUpdate(req.params.id, req.body, {new:
true, runValidators: true})
    if(!orders){
      return res.status(404).send()
    }
    res.send(orders)
  }
```

```
    }catch(e){
      res.status(400).send(e)
    }
  })
}
```

Delete Data of Orders with a unique Order ID

```
router.delete('/orders/:id', async (req, res) => {
  try{
    const orders = await Orders.findByIdAndDelete(req.params.id);
    console.log(orders)
    if(!orders){
      return res.status(404).send()
    }
    res.send(orders)
  }catch{
    res.status(500).send()
  }
})
```

Delivery Person

Add/post a new Delivery Person to database

```
router.post('/deliveryperson', (req, res) => {
  // res.header("Access-Control-Allow-Origin", "*");
  console.log(req.body)
  const deliveryperson = new DeliveryPerson(req.body)
  deliveryperson.save().then(() => {
    res.status(201)
    res.send(deliveryperson)
  }).catch((e) => {
    res.status(400)
    res.send(e)
    console.log(e)
  })
})
}
```

←

Name *

Arun Krishna

Gender

Male

Phone Number *

35456786543

Address Line 1 *

Vit Vellore Vit Vellore

Address Line 2

City *

Vellore

State *

Tamil Nadu

Country *

India

☒ Is Active

Save Delete

Get all Deliver Person Data






```
router.get('/deliveryperson', (req,res) => {

  res.header("Access-Control-Allow-Origin", "*");
  DeliveryPerson.find({}).then((deliveryperson) => {
    res.send(deliveryperson)
  }).catch((e) => {
    res.status(500)
    res.send()
  })
})
```

Manage Deliver Person

+ Add

Search

SNo	Name	Date Joined	Contact	Active?	Edit
1	Sivasanmugam	7/23/2020, 6:34:45 PM	4343434343	<input checked="" type="checkbox"/>	
2	Sivasanmugam	7/23/2020, 5:42:33 PM	4343434343	<input checked="" type="checkbox"/>	
3	Rohan MITTAL	7/23/2020, 9:31:24 PM	8754583601	<input checked="" type="checkbox"/>	
4	Rohan MITTAL	7/25/2020, 2:29:12 PM	8754583601	<input checked="" type="checkbox"/>	
5	Arun Krishna	10/10/2020, 10:49:32 AM	35456786543	<input checked="" type="checkbox"/>	

Items per page: 5

1 – 5 of 5

<<

<

>

>>

Get Data of Delivery Person with a unique Id

```
router.get('/deliveryperson/:id', async (req,res) => {
  res.header("Access-Control-Allow-Origin", "*");
  const _id = req.params.id
  try{
    const deliveryperson = await DeliveryPerson.findById(_id)
    if(!deliveryperson){
      return res.status(404).send()
    }
    res.status(200).send(deliveryperson)
  }catch(e){
    res.status(500).send(e)
  }
})
```

The image shows a web form for creating or updating a delivery person. The form fields are as follows:

- Name: Sivasanmugam
- Gender: Others
- Phone Number: 4343434343
- Address Line 1: Arunima Appt
- Address Line 2: Adyar
- City: Chennai
- State: Tamil Nadu
- Country: India
- Is Active: ☒

Buttons: Save, Delete

Below the form is a REST client showing a GET request to `localhost:3000/deliveryperson/5f197fd921fef62d77ce9d7f`. The response is a JSON object:

```
{
  "address": {
    "addressLine1": "Arunima Appt",
    "addressLine2": "Adyar",
    "city": "Chennai",
    "state": "Tamil Nadu",
    "country": "India"
  },
  "_id": "5f197fd921fef62d77ce9d7f",
  "name": "Sivasanmugam",
  "phoneNumber": 4343434343,
  "dateJoined": "2020-07-23T13:04:45.000Z",
  "isActive": true,
  "__v": 0,
  "gender": "Others",
  "modifiedBy": "rohanmittal01@gmail.com",
  "modifiedDate": "2020-07-25T09:00:30.000Z"
}
```


Update Data of Delivery Person with a unique Id

```
router.patch('/deliveryperson/:id', async (req, res) => {
  try{
    const deliveryperson = await DeliveryPerson.findByIdAndUpdate(req.params.id,
req.body, {new: true, runValidators: true})
    if(!deliveryperson){
      return res.status(404).send()
    }
    res.send(deliveryperson)
  }catch(e){
    res.status(400).send(e)
  }
})
```

Delete Data of Delivery Person with a unique Id (not used, but kept just in case)

```
router.delete('/deliveryperson/:id', async (req, res) => {
  try{
    const deliveryperson = await DeliveryPerson.findByIdAndDelete(req.params.id);
    console.log(deliveryperson)
    if(!deliveryperson){
      return res.status(404).send()
    }
    res.send(deliveryperson)
  }catch{
    res.status(500).send()
  }
})
```

Categories

Add/post a new Categories to database

```
router.post('/categories', (req, res) => {
  // res.header("Access-Control-Allow-Origin", "*");
  console.log(req.body)
  const category = new Category(req.body)
  category.save().then(() => {
    res.status(201)
    res.send(category)
  }).catch((e) => {
    res.status(400)
    res.send(e)
    console.log(e)
  })
})
```

←

Key IWP













Name IWP

Save Delete

Get all Category Data

```
router.get('/categories', (req,res) => {

    res.header("Access-Control-Allow-Origin", "*");
    Category.find({}).then((categories) => {
        res.send(categories)
    }).catch((e) => {
        res.status(500)
        res.send()
    })
})
```

SNo	Key	Name	Edit
1	bread	Bread	 
2	vegetables	Vegetables	 
3	fruits	Fruits	 
4	dairy	Dairy	 
5	seasonings and spices	Seasonings and Spices	 
6	IWP	IWP	 

Items per page: 10
1 - 6 of 6
|< < > >|

Choose Category

All Categories

Bread

Vegetables

Fruits

Dairy

Seasonings and Spices

IWP

(auto updating Categories Filter)

Get Data of Categories with a unique Id

```
router.get('/categories/:id', async (req,res) => {
  res.header("Access-Control-Allow-Origin", "*");
  const _id = req.params.id
  try{
    const category = await Category.findById(_id)
    if(!category){
      return res.status(404).send()
    }
    res.status(200).send(category)
  }catch(e){
    res.status(500).send(e)
  }
})
```

GET localhost:3000/categories/5f8145ac5e86db0a4c3dfb52

Params Authorization Headers (6) Body Pre-request Script Tests Settings

Body Cookies Headers (7) Test Results

Pretty Raw Preview Visualize JSON











```
1 {
2   "_id": "5f8145ac5e86db0a4c3dfb52",
3   "key": "IWP",
4   "name": "IWP",
5   "dateAdded": "2020-10-10T05:25:00.000Z",
6   "addedBy": "rohanmittal01@gmail.com",
7   "modifiedDate": "2020-10-10T05:25:00.000Z",
8   "modifiedBy": "rohanmittal01@gmail.com",
9   "__v": 0
10 }
```

Update Data of Categories with a unique Id

```
router.patch('/categories/:id', async (req,res) => {
  try{
    const category = await Category.findByIdAndUpdate(req.params.id, req.body,
    {new: true, runValidators: true})
    if(!category){
      return res.status(404).send()
    }
    res.send(category)
  }catch(e){
    res.status(400).send(e)
  }
})
```

Delete Data of Categories with a unique Id

```
router.delete('/categories/:id', async (req, res) => {
  try{
    const category = await Category.findByIdAndDelete(req.params.id);
    console.log(category)
    if(!category){
      return res.status(404).send()
    }
    res.send(category)
  }catch{
    res.status(500).send()
  }
})
```

SNo	Key	Name	Edit
1	bread	Bread	 
2	vegetables	Vegetables	 
3	fruits	Fruits	 
4	dairy	Dairy	 
5	seasonings and spices	Seasonings and Spices	 

Items per page: 10 1 – 5 of 5 |< < > >|

All Categories
Bread
Vegetables
Fruits
Dairy
Seasonings and Spices