

BIG BAZAR

### Product Details



Status: **Inactive**

**Mobile Tripod**

SKU: SKU007

**\$359**

In Stock QUANTITY: 200

Flexible tripod stand for mobile phones.

**Category:** Accessories

Total Price: \$359

[accessories](#) [mobile tripod](#) [new-accessories](#)

**Cart Details** X

 Gaming Mouse Pad Accessories 799	<input type="button" value="Delete"/>
 Bluetooth Headphones Electronics 1999	<input type="button" value="Delete"/>
 Mobile Tripod Accessories 359	<input type="button" value="Delete"/>

BIG BAZAR

[Categories](#) [About Us](#) [Contact Us](#) [Offers](#)   

### Product Details



Status: **Inactive**

**Wireless Mouse**

SKU: SKU001

**\$499**

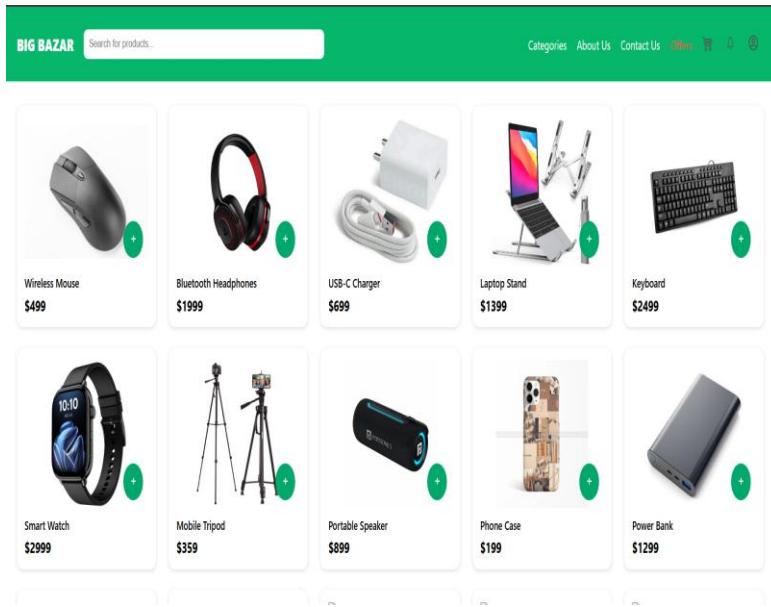
In Stock QUANTITY: 120

Ergonomic wireless mouse with USB receiver.

**Category:** Electronics

Total Price: \$499

[electronics](#) [wireless mouse](#) [new-electronics](#)



The screenshot shows the Dashtar dashboard with the 'Orders' section selected. The interface includes a sidebar with navigation links and a main table displaying order details.

**Orders**

INVOICE NO	ORDER TIME	CUSTOMER NAME	METHOD	AMOUNT	STATUS	ACTION
INV-1770111219434	2/3/2026, 3:03:39 PM	itg user	Online	\$799	Placed	Placed
INV-1770111207406	2/3/2026, 3:03:27 PM	itg user	Online	\$2698	Placed	Placed
INV-177009987333	2/3/2026, 11:56:27 AM	itg user	Online	\$2498	Placed	Placed

**Dashtar**

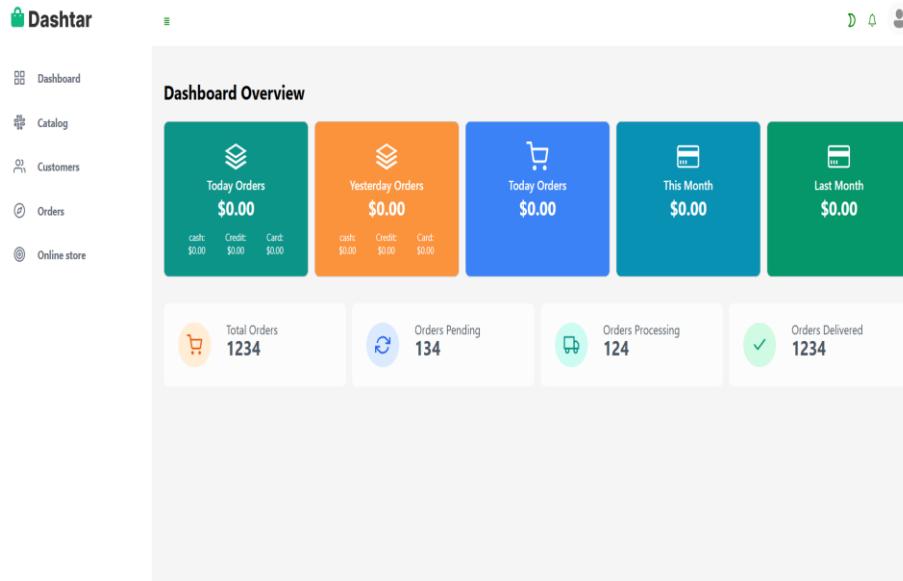
The screenshot shows the 'Customers' section of the Dashtar dashboard. It features a search bar at the top labeled 'Search by name/email'. Below the search bar is a table with columns: S.NO, JOINING DATE, NAME, EMAIL, ROLE, and ACTIONS. The table contains 9 rows of customer data. Each row includes a small profile icon, the customer's name, their email address, and their role as 'Customer'. The 'ACTIONS' column for each row contains three icons: a pencil, a magnifying glass, and a trash can.

S.NO	JOINING DATE	NAME	EMAIL	ROLE	ACTIONS
1	November 15, 2025	mayuri rajput	mayurirajput@itgeeks.com	Customer	
2	November 15, 2025	itg user	itguser@gmail.com	Customer	
3	November 15, 2025	rohit sharma	rohitsharma@itgeeks.com	Customer	
4	November 15, 2025	mohit singh	mohitsingh@itgeeks.com	Customer	
5	November 15, 2025	rohit thakur	rohitthakur@itgeeks.com	Customer	
6	November 15, 2025	anuj malakar	anuj@itgeeks.com	Customer	
7	November 15, 2025	atul shukla	atulshukla@itgeeks.com	Customer	
8	November 15, 2025	newuser	aaa234@itg.com	Customer	
9	February 3, 2026	Mayuri Rajput	mayuri123@itg.com	Customer	

**Dashtar**

The screenshot shows the 'Products' section of the Dashtar dashboard. It features a search bar at the top labeled 'Search Product'. Below the search bar is a table with columns: PRODUCT NAME, CATEGORY, PRICE, SALE PRICE, STOCK, STATUS, VIEW, PUBLISHED, and ACTIONS. The table contains 7 rows of product data. Each row includes a small product thumbnail, the product name, its category, price, sale price, stock level, and status. The 'STATUS' column for all products shows 'Selling'. The 'PUBLISHED' column has a green switch that is turned on for all products. The 'ACTIONS' column for each row contains three icons: a pencil, a magnifying glass, and a trash can.

PRODUCT NAME	CATEGORY	PRICE	SALE PRICE	STOCK	STATUS	VIEW	PUBLISHED	ACTIONS
Wireless Mouse	Electronics	\$499.00	\$499.00	120	<span>Selling</span>			
Bluetooth Headphones	Electronics	\$1999.00	\$1999.00	80	<span>Selling</span>			
USB-C Charger	Electronics	\$699.00	\$699.00	150	<span>Selling</span>			
Laptop Stand	Accessories	\$1399.00	\$1399.00	60	<span>Selling</span>			
Keyboard	Electronics	\$2499.00	\$2499.00	40	<span>Selling</span>			
Smart Watch	Electronics	\$2999.00	\$2999.00	70	<span>Selling</span>			
Mobile Tripod	Accessories	\$359.00	\$359.00	200	<span>Selling</span>			



File Edit Selection View Go Run Terminal Help

EXPLORER

FULL-STACK-WEBSITE

- my-app
  - backend
  - build
  - frontend
  - public
  - src
    - assets
    - components
    - pages
      - App.js
      - AppTest.js
      - index.css
      - index.js
      - logo.svg
      - reportWebVitals.js
      - setupTests.js
      - gitignore
      - e-commerce-website.pdf
    - package-lock.json
    - package.json
    - README.md
  - public
  - svr
  - env
  - node\_modules
  - package-lock.json
  - package.json
  - README.md

OUTLINE

TIMELINE

MainGrid.jsx M orders.jsx U order.jsx U productGrid.jsx productDetails.jsx env register.css

```

product.jsx
1 import React from 'react';
2 import './assets/main.css'
3 export default function MainGrid() {
4   return (
5     <div>
6       <h2>Dashboard Overview</h2>
7       <div>
8         <div>
9           <div>
10             <div>
11               <div>
12                 <div>
13                   <div>
14                     <div>
15                       <div>
16                         <div>
17                           <div>
18                             <div>
19                               <div>
20                                 <div>
21                                   <div>
22                                     <div>
23                                       <div>
24                                         <div>
25                                           <div>
26                                             <div>
27                                               <div>
28                                                 <div>
29                                                   <div>
30                                                     <div>
31                                                       <div>
32             </div>
33           </div>
34         </div>
35       </div>
36     </div>
37   </div>
38 
```

my-app > frontend > App.js

```

1 import { MdDashboard } from 'react-icons/md';
2 import './App.css';
3 import {
4   BrowserRouter as Router,
5   Routes,
6   Route,
7   Navigate,
8   Outlet,
9 } from "react-router-dom";
10 import Dashboard from './pages/dashboard';
11 import Login from './pages/login';
12 import Register from './pages/register';
13 import Customer from './pages/customer';
14 import MainGrid from './components/MainGrid';
15 import Catalog from './pages/catalog';
16 import CatalogDetails from './components/catalogDetails';
17 import Product from './pages/product';
18 import OnlineStore from './pages/onlineStore';
19 import ProductCart from './components/ProductCart';
20 import ProductDetails from './components/ProductCart/productDetails';
21 import Orders from './pages/orders';
22 function App() {
23   return (
24     <Router>
25       <Routes>
26         <Route path="/login" element={<Navigate to="/login"/>} />
27         <Route path="/login" element={<Login />} />
28         <Route path="/register" element={<Register />} />
29         /* Dashboard layout wrapper */
30         <Route path="/dashboard" element={<Dashboard role="admin"/>} />
31         /* Default Dashboard content */
32         <Route index element={<MainGrid />} />
33         <Route path="/" element={<MainGrid />} />
34         <Route path="customers" element={<Customer />} />
35         <Route path="catalog" element={<Catalog />} />
36         <Route path="catalog/:id" element={<CatalogDetails />} />
37         <Route path="order" element={<Orders />} />
38       </Routes>
39       <Route path="/" element={<Layout wrapped />} />
40       <Route path="/onlinestore" element={<OnlineStore />} />
41       <Route index element={<ProductGrid />} />
42       <Route path="/product/:id" element={<ProductDetails />} />
43     </Router>
44 
```

FULL-STACK-WEBSITE

```

my-app > frontend > components > Navbar.js > Navbar
  1 import React from "react";
  2 import "./assets/main.css";
  3 import logo from "./assets/images/logo.svg";
  4 export default function Navbar({ role }) {
  5   return (
  6     <div className="navbar-container">
  7       <div className="navbar-container-logo">
  8         <img src={logo}>/<img/>
  9       </div>
 10      <div className="navbar-container-list">
 11        {role == "admin" && <a href="/dashboard">
 12          <div>
 13            <img stroke="currentColor" fill="none" stroke-width="2" viewBox="0 0 24 24" stroke-linecap="round" stroke-linejoin="round"/>
 14            Dashboard</div>
 15        </a>}
 16        <a href="/dashboard/catalog">
 17          <div>
 18            <img stroke="currentColor" fill="none" stroke-width="2" viewBox="0 0 24 24" stroke-linecap="round" stroke-linejoin="round"/>
 19            Catalog</div>
 20        </a>
 21        <a href="/dashboard/customers">
 22          <div>
 23            <img stroke="currentColor" fill="none" stroke-width="2" viewBox="0 0 24 24" stroke-linecap="round" stroke-linejoin="round"/>
 24            Customers</div>
 25        </a>
 26        <a href="/dashboard/order">
 27          <div>
 28            <img stroke="currentColor" fill="none" stroke-width="2" viewBox="0 0 24 24" stroke-linecap="round" stroke-linejoin="round"/>
 29            Order</div>
 30        </a>
 31        <a href="/customer/product">
 32          <div>
 33            <img stroke="currentColor" fill="none" stroke-width="2" viewBox="0 0 24 24" stroke-linecap="round" stroke-linejoin="round"/>
 34            Product</div>
 35        </a>
 36      </div>
 37    </div>
 38  )
 39 }
 40 
```

FULL-STACK-WEBSITE

```

my-app > backend > services > cartService.js > deleteCartData
  1 import Product from "../models/Product.js";
  2 import Cart from "../models/Cart.js";
  3 
  4 export const addtocart = async ({customer_id, product_id, quantity, price}) => {
  5   const product = await Product.findById(product_id);
  6   if (!product) {
  7     throw new Error("Product not found!");
  8   }
  9   // Prepare snapshot
 10   const productsnapshot = {
 11     product_id: product._id,
 12     product_name: product.name,
 13     description: product.description,
 14     price: product.price,
 15     category: product.category,
 16     stock_quantity: product.stock_quantity,
 17     sku: product.sku,
 18     weight_kg: product.weight_kg,
 19     item_id: product._id,
 20     product_img: product.product_img,
 21     status: product.status,
 22     quantity,
 23   };
 24   let cart = await Cart.findOne({ customer_id });
 25   if (!cart) {
 26     const newcart = await Cart.create({
 27       customer_id,
 28       items: [productsnapshot],
 29     });
 30   } else {
 31     const existingitem = cart.items.find(
 32       item => item.product_id === product_id
 33     );
 34     if (existingitem) {
 35       existingitem.quantity += quantity;
 36       existingitem.price += price;
 37     } else {
 38       cart.items.push(productsnapshot);
 39     }
 40   }
 41   await cart.save();
 42 }
 43 
```

FULL-STACK-WEBSITE

```

my-app > backend > controllers > orderController.js > getOrders > orders
  1 import { createOrderData, getAllOrderservice } from "../services/orderService.js";
  2 
  3 export const createorder = async (req, res) => {
  4   try {
  5     const orderdata = req.body;
  6     const order = await createOrderData(orderdata);
  7     return res.status(201).json({
  8       success: true,
  9       message: "Order created successfully",
 10       data: order,
 11     });
 12   } catch (error) {
 13     console.error("Create Order Error:", error);
 14   }
 15   return res.status(500).json({
 16     success: false,
 17     message: error.message || "Failed to create order",
 18   });
 19 }
 20 
```

```

 21 
```

```

 22 
```

```

 23 
```

```

 24 
```

```

 25 
```

```

 26 
```

```

 27 
```

```

 28 
```

```

 29 
```

```

 30 
```

```

 31 
```

```

 32 
```

```

 33 
```

```

 34 
```

```

 35 
```

```

 36 
```

```

 37 
```

```

 38 
```

```

 39 
```

```

 40 
```

The screenshot shows the Explorer view of a code editor with the following file structure:

- FULL-STACK-WEBSITE
- my-app
- backend
- controllers
- controller
- exports
- middleware
- models
- node\_modules
- routes
- schemas
- services
- utils
- validations
- env
- gitignore
- index.js
- index.router.js
- package.json
- pid
- renderv
- public
- src
- env
- README.md
- node\_modules
- package-lock.json

The `index.js` file content is as follows:

```
const express = require('express');
const dotenv = require('dotenv');
const cors = require('cors');
const connection = require('./utils/db');
const productRouter = require('./routes/productRoutes.js');
const cartRouter = require('./routes/cartRoutes.js');
const orderRouter = require('./routes/orderRoutes.js');

dotenv.config();

const app = express();
app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));
app.use('/api/products', productRouter);
app.use('/api/cart', cartRouter);
app.use('/api/order', orderRouter);

const port = process.env.PORT || 2000;
app.listen(port, () => {
  console.log(`Server started on port ${port}`);
});
```

The screenshot shows the `Product.js` file in the `models` directory. The code defines a Mongoose schema for a `Product`.

```
import mongoose from "mongoose";

const productSchema = new mongoose.Schema({
  product_id: { ... },
  product_name: { ... },
  description: { ... },
  price: { ... },
  category: { ... },
  stock_quantity: { ... },
  sku: { ... },
  weight_kg: { ... },
  is_active: { ... },
  // NEW FIELD - product image
  product_img: { ... },
  // ★ NEW FIELD - product status
  status: { ... },
  { timestamps: true }
});

export default mongoose.model("Product", productSchema);
```