



BOOB AADY

An e-commerce application

By:

Ilnaz Alizadeh

Hossein Parsa

Amir Sharifi

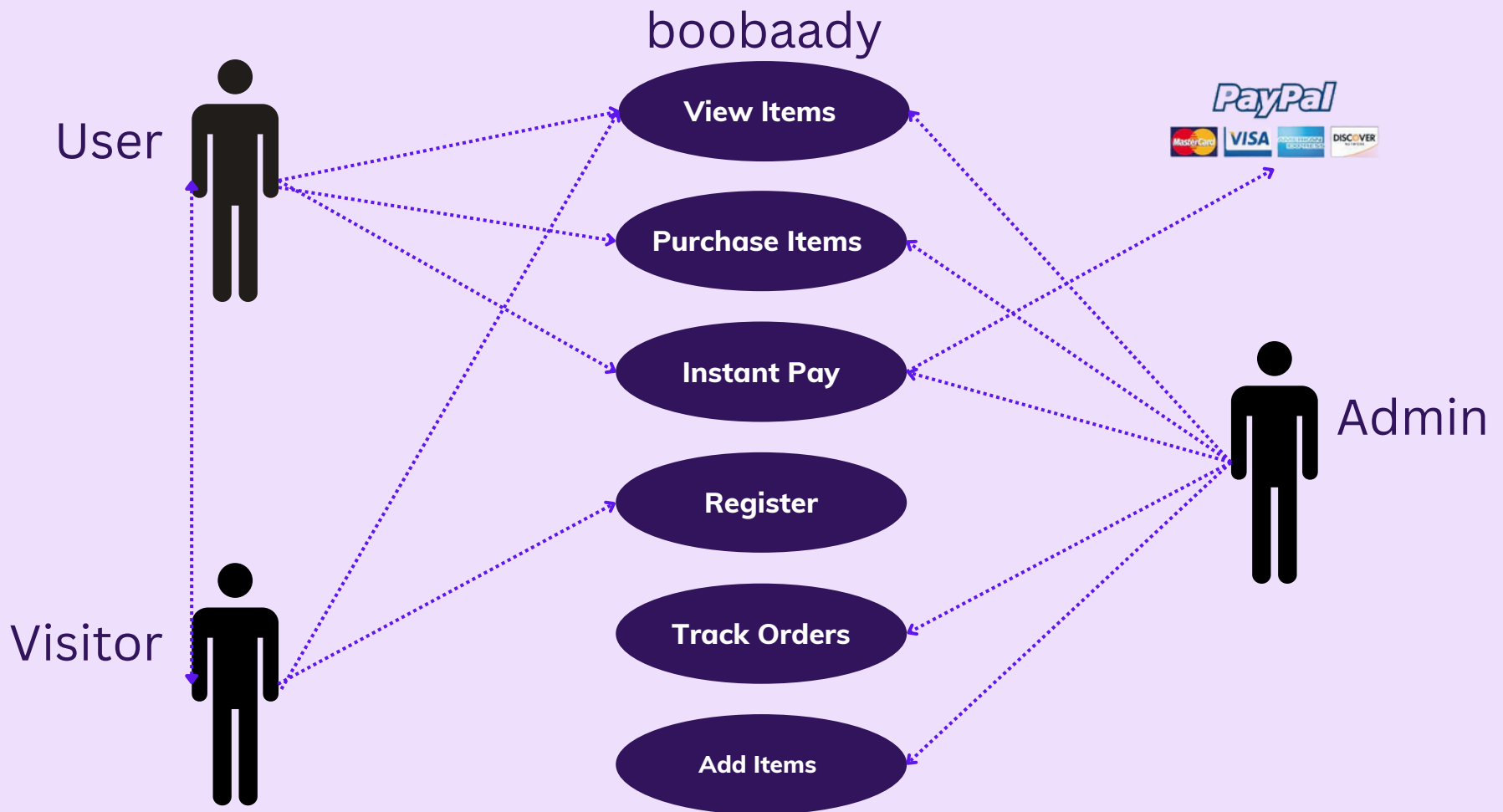
e-commerce

- What is e-commerce?
- Types of e-commerce.
- E-commerce applications.



Our solutions

Use-case Diagram



Our solutions

Admin panel

- Login
- Update profile

Admin Profile

Name

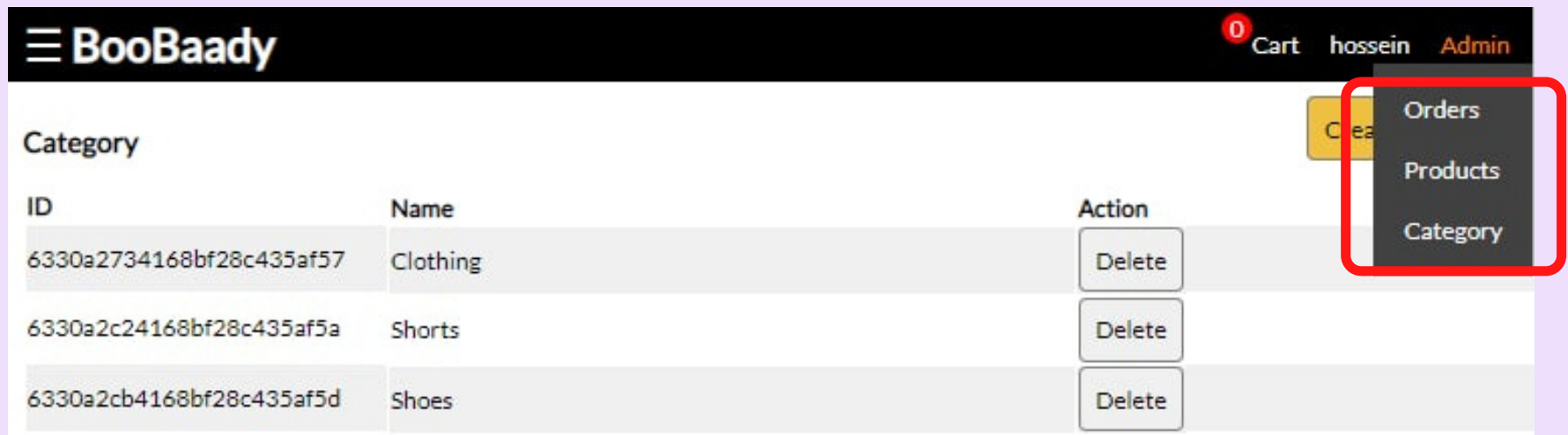
Email

Password

Our solutions

Admin panel

- Add/Remove category
- See the list of orders



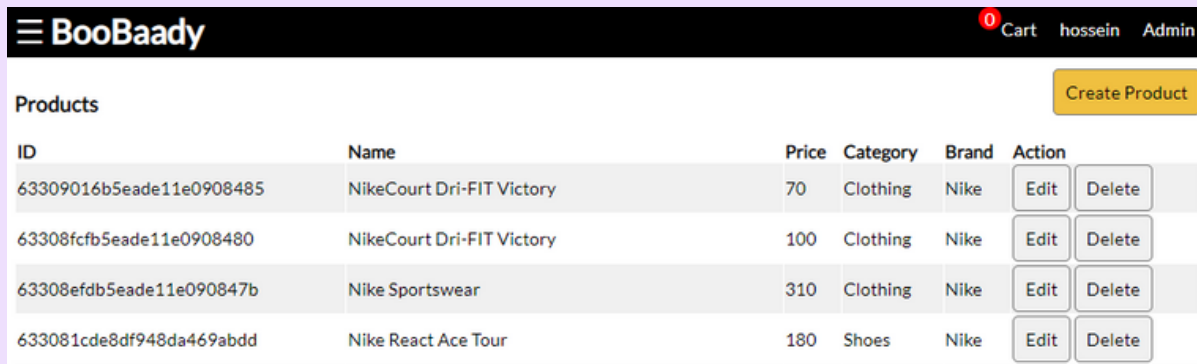
The screenshot displays the BooBaady Admin Panel. At the top, there is a navigation bar with the BooBaady logo, a user profile for 'hossein', and links for 'Cart' and 'Admin'. A dropdown menu is open, showing 'Orders', 'Products', and 'Category' options, which are highlighted with a red box. Below the navigation bar, there is a table titled 'Category' with columns for 'ID', 'Name', and 'Action'. The table contains three rows of category data, each with a 'Delete' button.

ID	Name	Action
6330a2734168bf28c435af57	Clothing	Delete
6330a2c24168bf28c435af5a	Shorts	Delete
6330a2cb4168bf28c435af5d	Shoes	Delete

Our solutions

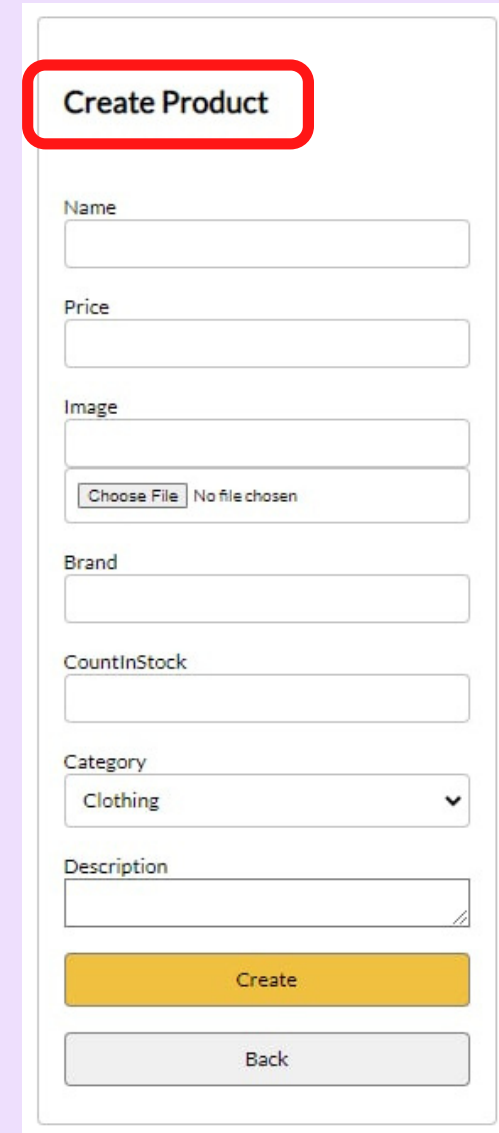
Admin panel

- Add/Remove/Edit product



The screenshot shows the BooBaady Admin Panel. At the top, there is a navigation bar with the BooBaady logo, a cart icon with a red notification bubble containing the number '0', and the user name 'hossein Admin'. Below the navigation bar, there is a 'Products' section with a 'Create Product' button. The main content is a table with the following data:

ID	Name	Price	Category	Brand	Action
63309016b5eade11e0908485	NikeCourt Dri-FIT Victory	70	Clothing	Nike	Edit Delete
63308fcb5eade11e0908480	NikeCourt Dri-FIT Victory	100	Clothing	Nike	Edit Delete
63308efdb5eade11e090847b	Nike Sportswear	310	Clothing	Nike	Edit Delete
633081cde8df948da469abdd	Nike React Ace Tour	180	Shoes	Nike	Edit Delete



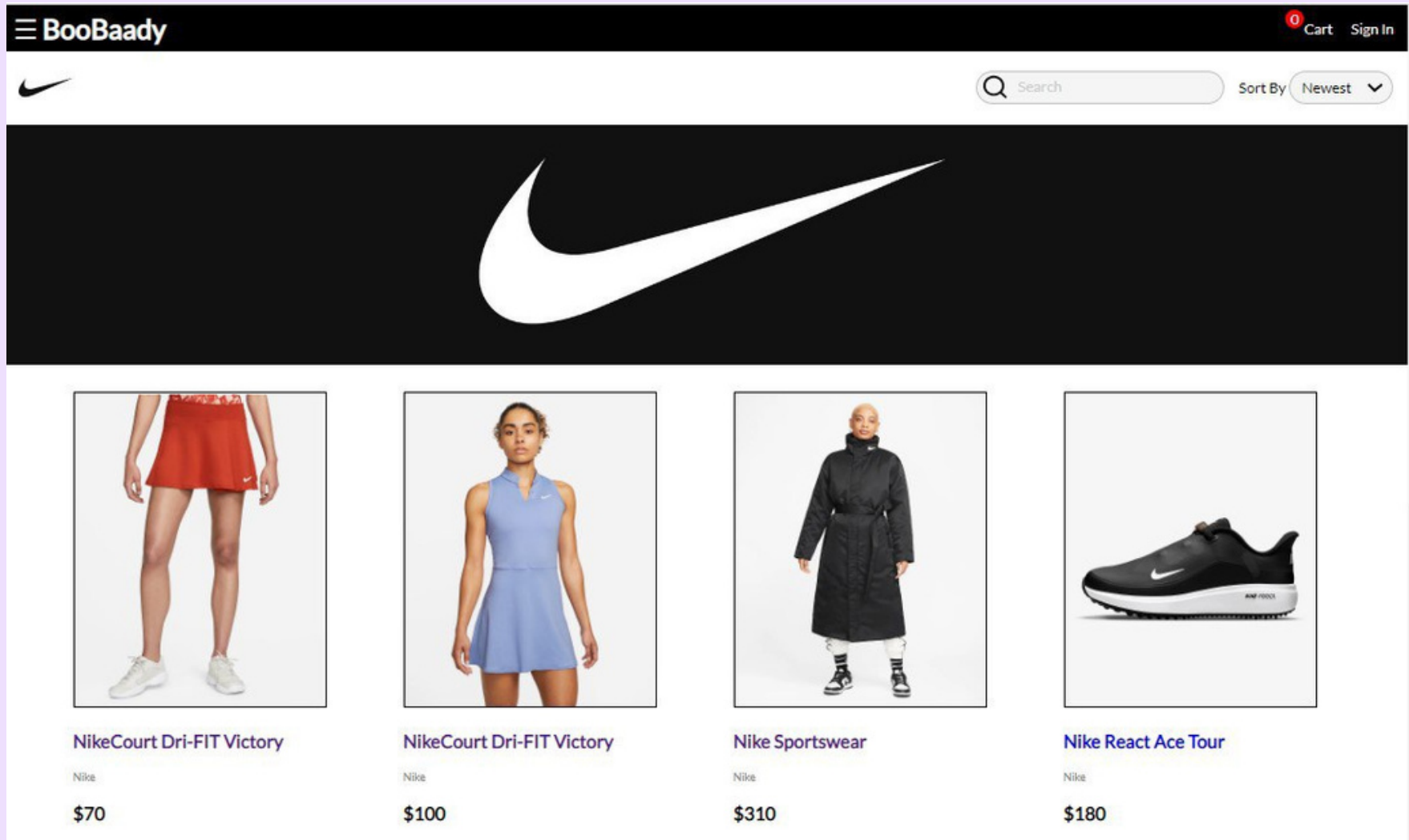
The screenshot shows the 'Create Product' form. The form is titled 'Create Product' and contains the following fields:

- Name**: A text input field.
- Price**: A text input field.
- Image**: A text input field with a 'Choose File' button and the text 'No file chosen'.
- Brand**: A text input field.
- CountInStock**: A text input field.
- Category**: A dropdown menu with 'Clothing' selected.
- Description**: A text area with a diagonal slash icon in the bottom right corner.

At the bottom of the form, there are two buttons: a yellow 'Create' button and a grey 'Back' button.

Our solutions

User Activities



The screenshot displays the Nike website interface. At the top left, the 'BooBaady' logo is visible. The top right corner features a 'Cart' icon with a red notification bubble and a 'Sign In' link. Below the navigation bar is a search bar with a magnifying glass icon and the text 'Search', and a 'Sort By' dropdown menu set to 'Newest'. The main content area is dominated by a large black banner with the white Nike swoosh logo. Below the banner, four product listings are shown in a grid:

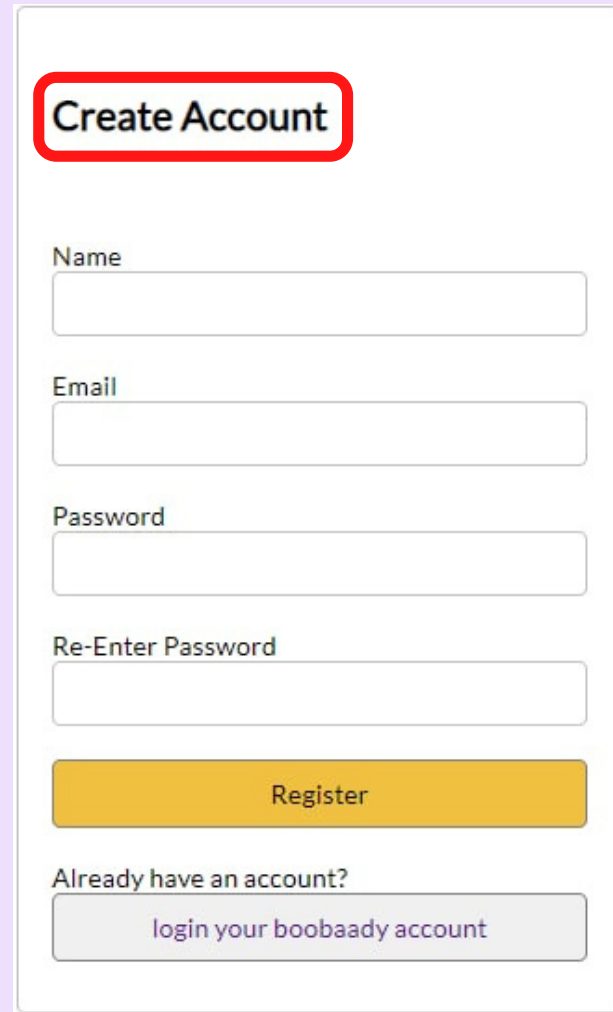
- Product 1:** NikeCourt Dri-FIT Victory (Red shorts). Price: \$70.
- Product 2:** NikeCourt Dri-FIT Victory (Blue dress). Price: \$100.
- Product 3:** Nike Sportswear (Black coat). Price: \$310.
- Product 4:** Nike React Ace Tour (Black and white sneaker). Price: \$180.

Each product listing includes a small image of the item, the product name, the Nike brand name, and the price.

Our solutions

User Activities

- Registration
- Login
- Update profile



The image shows a user registration form with a white background and a thin grey border. At the top, the text "Create Account" is enclosed in a red rounded rectangle. Below this are four input fields: "Name", "Email", "Password", and "Re-Enter Password". Each field is a simple white rectangle with a thin grey border. Below the "Re-Enter Password" field is a yellow "Register" button. At the bottom, there is a link "Already have an account?" with a grey button containing the text "login your boobaady account".

Create Account

Name

Email

Password

Re-Enter Password

Register

Already have an account?

Our solutions

User Activities

- Add/Remove product to cart and proceed to checkout
- Add comment and rating

NikeCourt Dri-FIT Victory

Price: **\$100**

Description:
Women's Tennis Dress

Price: 100

Status: In Stock

Qty:

Add to Cart

checkout

Reviews

There is no review

Write a customer review

Rating

Comment

Submit

Our solutions

User Activities

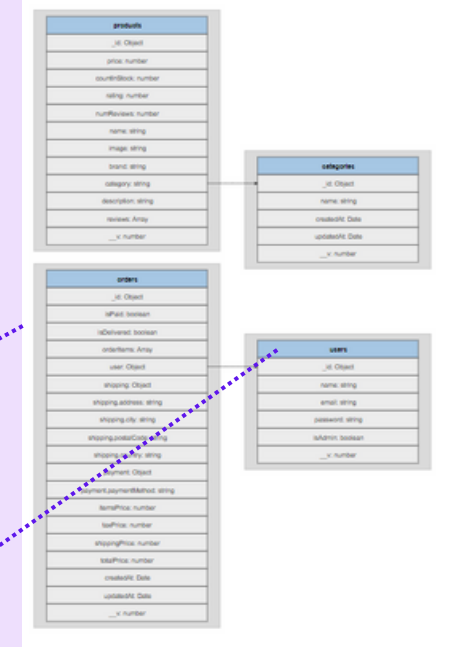
- Add shipping address
- Make payment

<p>Shipping</p> <p>803apt,2500 peel, Montreal,H5B 2R2, Canada, Not Delivered.</p>	<p>PayPal</p> <p><input type="checkbox"/> Debit or Credit Card</p> <p>Powered by PayPal</p>								
<p>Payment</p> <p>Payment Method: paypal Not Paid.</p>	<p>Order Summary</p> <table border="0"> <tr> <td>Items</td> <td>\$48</td> </tr> <tr> <td>Shipping</td> <td>\$10</td> </tr> <tr> <td>Tax</td> <td>\$7.2</td> </tr> <tr> <td>Order Total</td> <td>\$65.2</td> </tr> </table>	Items	\$48	Shipping	\$10	Tax	\$7.2	Order Total	\$65.2
Items	\$48								
Shipping	\$10								
Tax	\$7.2								
Order Total	\$65.2								
<p>Shopping Cart</p> <hr/> <p>Cart is empty</p> <p style="text-align: right;">Price</p>									

DataBase

orders
_id: Object
isPaid: boolean
isDelivered: boolean
orderItems: Array
user: Object
shipping: Object
shipping.address: string
shipping.city: string
shipping.postalCode: string
shipping.country: string
payment: Object
payment.paymentMethod: string
itemsPrice: number
taxPrice: number
shippingPrice: number
totalPrice: number
createdAt: Date
updatedAt: Date
__v: number

users
_id: Object
name: string
email: string
password: string
isAdmin: boolean
__v: number



DataBase

products
_id: Object
price: number
countInStock: number
rating: number
numReviews: number
name: string
image: string
brand: string
category: string
description: string
reviews: Array
__v: number

categories
_id: Object
name: string
createdAt: Date
updatedAt: Date
__v: number

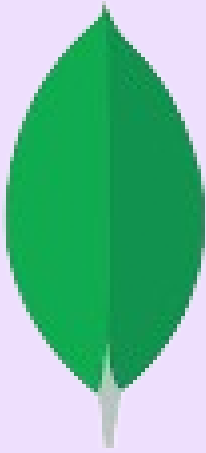
products
_id: Object
price: number
countInStock: number
rating: number
numReviews: number
name: string
image: string
brand: string
category: string
description: string
reviews: Array
__v: number

categories
_id: Object
name: string
createdAt: Date
updatedAt: Date
__v: number

orders
_id: Object
productId: Object
quantity: number
total: number
status: string
user: Object
shipping: Object
shippingAddress: string
shippingCity: string
shippingPostalCode: string
shippingCountry: string
payment: Object
paymentPaymentMethod: string
itemPrice: number
taxPrice: number
shippingPrice: number
totalPrice: number
createdAt: Date
updatedAt: Date
__v: number

users
_id: Object
name: string
email: string
password: string
isAdmin: boolean
__v: number

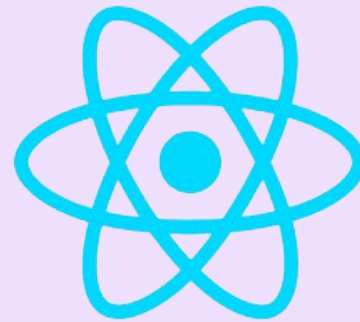
Technology used



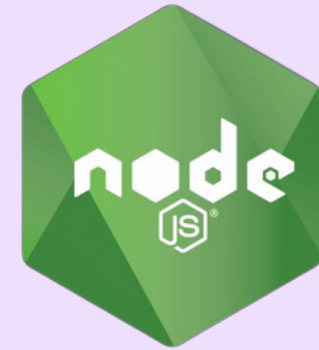
M

express

E



R



N

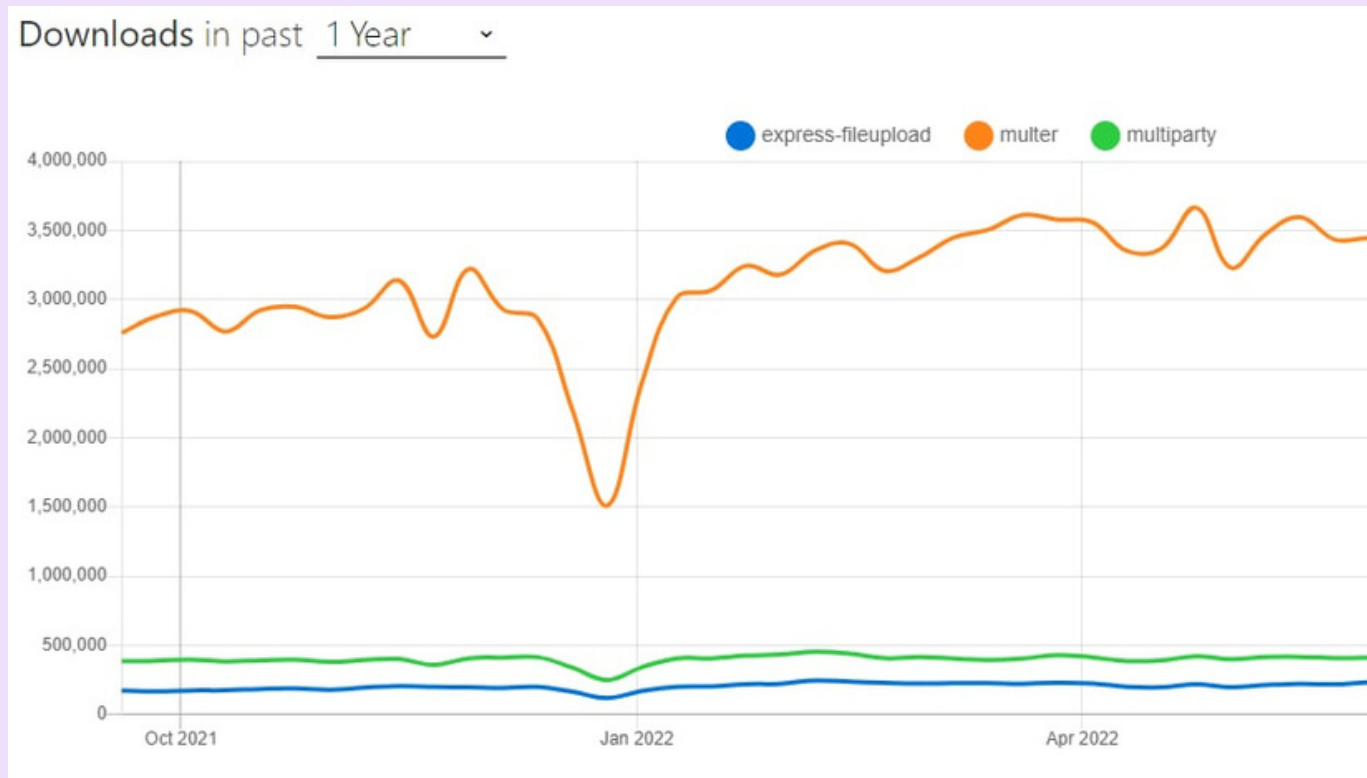
Major problems

- How to manage the workflow of the project?
- How to design UI of the project?
- what data base to use?
- How to deal with mongo collections instead of MySql tables?

What we learned

How to upload a file

Multer package



What we learned

How to upload a file

Multer package

```
const storage = multer.diskStorage({
  destination(req, file, cb) {
    cb(null, 'frontend/public/uploads');
  },
  filename(req, file, cb) {
    cb(null, `${Date.now()}.jpg`);
  },
});
const upload = multer({ storage });
const router = express.Router();
router.post('/', upload.single('image'), (req, res) => {
  res.send(`/${req.file.path}`);
});
```


What we learned

How to handle JWT token and store user info on frontend client

- Solution: Use local storage

```
axios.post("/api/users/signin", { email, password }).then(res => {  
  localStorage.setItem('userInfo', JSON.stringify(res.data))  
  localStorage.setItem('token', res.data.token)  
  props.history.push(redirect);  
})
```

Syntax: `localStorage.setItem(key,value)`

What we learned

How to handle JWT token and store user info on frontend client

```
const placeOrderHandler = () => {
  axios.post("/api/orders", {cartItems, shipping, payment, itemsPrice,
    shippingPrice, taxPrice, totalPrice}, {
    headers: {
      Authorization: ' Bearer ' + localStorage.getItem('token')
    }
  }).then(res => {
    props.history.push("/order/" + res.data.data._id);
  })
}
```

Syntax: `localStorage.getItem(key)`

What we learned

How to use Hooks (useEffect)

Page doesn't render after changing a parameter

```
useEffect(() => {  
  |   |  getProducts();  
}, [category, sortOrder]);
```



What we learned

How to use Hooks (useEffect)

```
useEffect(() => {  
  |   getCategories()  
}, [])
```

Syntax:

```
useEffect(() => {  
  //Runs on the first render  
  //And any time any dependency value  
  changes  
}, [prop, state]);
```

Future work

- **User can signup / login using their social media accounts**
- **UI modification (more user friendly)**
- **More advanced admin pannel**
- **Add more options like
return and discount coupon**

Summary

- what was done

 - Chose the subject

 - Did research and made decision about technologies

 - Planned the work flow and divided the tasks

 - Examined different solutions for each task

- what is the result

 - Got one step closer to understand how a complete application works

 - Got better in doing research and solution finding