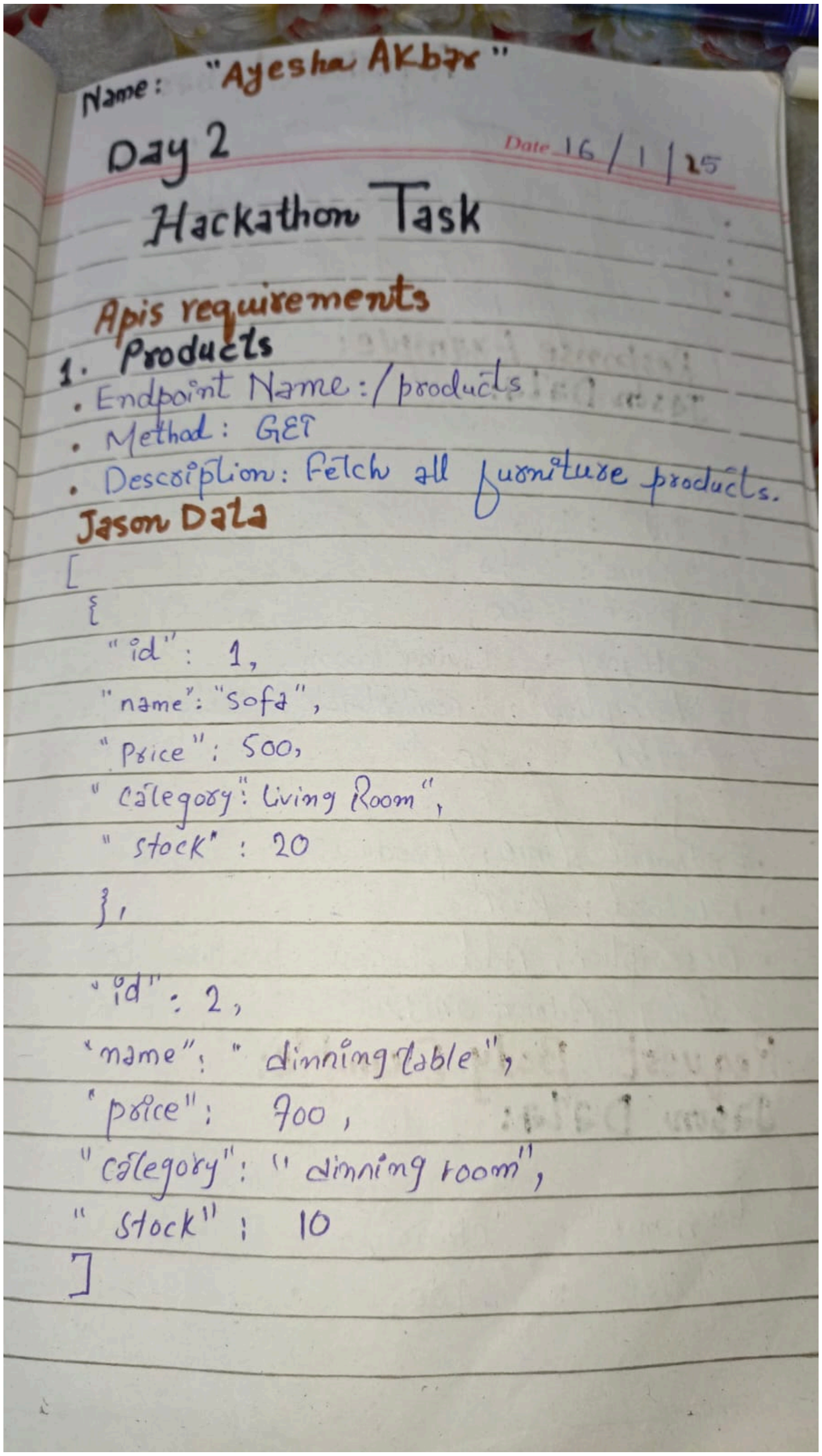
General Ecommerce website 2025



Name: "Ayesha Akbar"

Day 2 Date 16/1/25

Hackathon Task

Apis requirements

1. Products
   - Endpoint Name: /products
   - Method: GET
   - Description: Fetch all furniture products.

Jason Data

```
[
 {
  "id": 1,
  "name": "sofa",
  "Price": 500,
  "category": Living Room",
  " stock" : 20
 },
 "id": 2,
"name": " dinning table",
"price": 700 ,
"category": " dinning room",
" stock" : 10
]
```

2 "Ayesha Akbar"

"Day2 Hackathon Task" Date 18.1.25

- Endpoint Name: /Products/{id}
- Method: GET
- Description: fetch details of a specific furniture product by its ID.

Response Example:

Jason Data:

```
{
 "id" : 1,
 "name": "sofa",
 "price": 500,
 "category": " Living Room",
 "description": " comfortable 3-seater sofa",
 "stock" : 20
 },
```

- Endpoint Name: /products
- Method: POST
- Description: Add a new product to-the catalog (Admin only).

Request Body Example:

Jason Data:

```
{"name" : "Chair",
 "price" : 100,
```

Signature

No.

"category": "Living Room",
" description": "Ergonomic office chair",
"stock": 50
},

2. Categories
- Endpoint Name: / categories
- Method: GET
- Description: Fetch all product categories.

Response Example:
Jason Data:

```
[
 "Living Room", "Bed room", "Dining Room",
 "office", "Outdoor"
]
```

Endpoint Name: /categories/{id}/products

- Method: GET
- Description: Fetch products belonging to a
- specific category.

Response Example:
Jason Data: [

```
{
 "id" : 1,
```

"Ayesha Akbar"

"Day 2 Hackathon Task" Date 16·1·25

```
"name": "sofa",
" price" : 500,
" stock": 20,
},
{ "id" : 3,
"name" : "coffee Table",
" price " : 150
'stock" : 30
}
]
```

3· Users

- Endpoint Name: /users
- Method: POST
- Description: Register a new user.

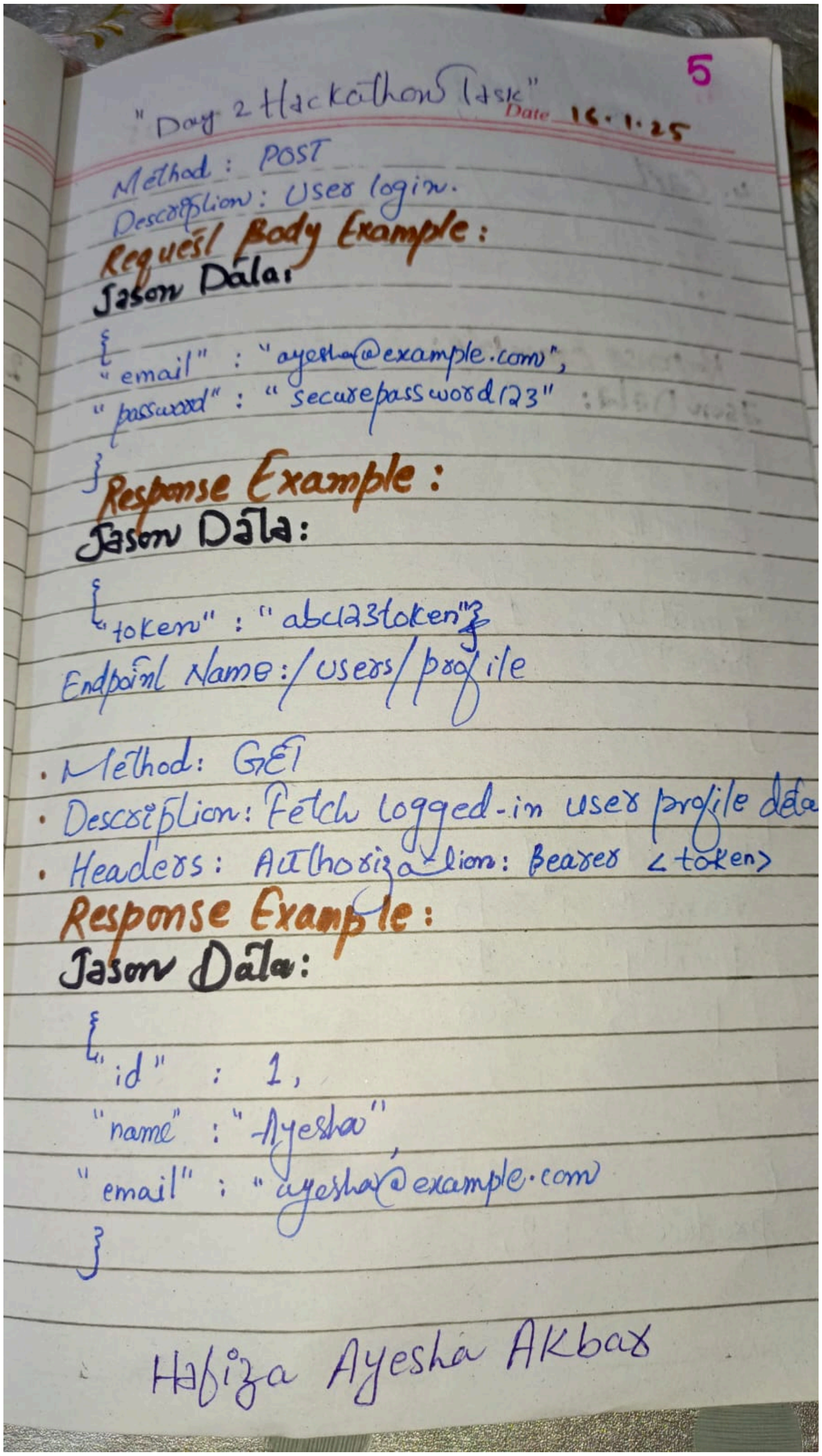
Request Body Example:

Jason Data:

```
{
" name": "Ayesha",
" email" : "ayesha@example.com",
"password" : " securepassword123"
}
```

Endpoint Name: /users/login

Signature ______ RC Ayesha Akbar



"Day 2 Hackathon Task" Date 16.1.25

Method: POST
Description: User login.

Request Body Example:

Jason Data:

```
{
"email" : "ayesha@example.com",
"password" : "securepassword123"
}
```

Response Example:

Jason Data:

```
{
"token" : "abc123token"}
```

Endpoint Name: /users/profile

- Method: GET
- Description: Fetch logged-in user profile dta
- Headers: Authorization: Bearer <token>

Response Example:

Jason Data:

```
{
"id" : 1,
"name" : "Ayesha",
"email" : "ayesha@example.com
}
```

Hafiza Ayesha Akbar

"Day 2 Hackathon Task" Date 16·1·25

4. Cart

Endpoint Name: /cart

- Method : GET
- Description: Fetch the current user's cart items.

Response Example:

Json Data:

```
[
  {
"productID" : 1,
"name"   : "sofa",
 "quantity" :   1,
 "price"    :   500
  },

 {
 "productId" : 1,
  "name"   : "sofa"
  "quantity"  :   1,
   "price"   : 500
},

 {
"productId" : 2,
```

Signature

" Day 2 Hackathon Task " Date 16.1.25

"name" : "Chair",
"quantity : 2,
"price" : 100
},
]

Endpoint Name: /cart
· Method : POST
· Description : Add an item to the cart.

Request Body Example:
Json Data:

```
{
"productId" : 1,
"quantity" : 1,
}
```

Endpoint Name: /cart/ {productId}

· Method : DELETE
· Description: Remove a product from the cart by product ID.

5. Orders
Endpoint Name : /orders
· Method : GET
· Description: Fetch a user's order history

"Day 2 Hackathon Task" Date 16.1.2

Response Example:

Json Data:

```
{
"orderId": 101,
"date" : "2025-01-16"
"total" : 600,
},
{
"orderId": 102,
"date" : 2025-01-11
"total" : 300
},
```

Endpoint Name: /orders

- Method: POST
- Description: Place a new order.

Request Body Example:

json Data:

```
{ "cartItems":
[{ "productId": 1,
  "quantity": 1},
{ "productId": 2,
 "quantity" : 2}],
```

Signature______

" Day 2 Hackathon(task" Date 16.1.25

"shippingAddress" : "123 Main st, country"}
}

6. **Reviews**
Endpoint Name: /products/{id}/reviews
• Method: GET
• Description: Fetch all reviews for a specific product.

json data:

```
[
{ "reviewId" : 1,
  "userId"   : 2,
  "rating"   : 5,
  "comment"  : "very comfortable sofa".
}
]
```

Endpoint Name: /products/{id}/reviews
• Method: POST
• Description: Add a new review for a product

Request Body Example:
json data:

```
{
"rating" : 5,
"comment" : "Amazing product, highly recommend!"
}
```

"Day 2 Hackathon Task" Date 16.1.25

7. Search.

Endpoint Name: /search
- Method: GET
- Description: Search for products by Keyword or filter criteria (eg. price range, categ

Query Parameters: ?
query= sofa and category = living-room and minprice= 200 and maxPrice = 1000

Response Example:

json data:

```
[
 {
 "id": 1,
 "name": "sofa",
 "price": 500,
 "stock": 20
 }
 ]
```

" Day 2 Hackathon Task "

11

Date 16.1.25

How the Frontend Interacts with Sanity CMS And Third-Party APIs. Describe

1. Frontend (React/Next.js):

Fetch product data (eg. name, price, stock, images) from Sanity CMS via the /products API.

Allow users to create orders through the /orders API.

Display shipment tracking details by interacting with the /shipment API.

2. Backend (Node.js / Express):

Handles API request from the frontend. Integrates with Sanity CMS for product management & order storage

"Day 2 Hackathon Task" Date 16.1./25

Fetches shipment tracking details from third-party APIs.

3. **Sanity CMS:**
Stores product details, categories and order data.
: Acts As the primary data Source for products and orders.

4. **Third-Party APIs:**
Provides real-time Shipment tracking information.

Signature

No.

"Day 2 Hackathon Task" Date 16.1.25

Key Workflow (short Version)

1. User Browses Products

Frontend: Sends GET/Products request to fetch product data.

Backend: Retrieves data from sanity CMS and returns it.

Frontend: Displays products.

2. User Adds Products to Cart

Frontend: Updates cart state locally (or sends to backend) :

Frontend: Shows Updated cart with Selected products and subtotal.

3. User Place an Order

Frontend: Sends post /orders with customer and product details.

Day 2 Hackathon Task

Date 16.1.

Backend: Displays order confirm

4. User Track Shipment

Frontend: Sends GET / Shipment ?orderId = 012345 to backend.

Backend: Retrieves Shipment status from third-party API.

Frontend: Displays shipment status and expected delivery.

5. Admin Adds/Updates Product

frontend: Sends POST / Products with product details.

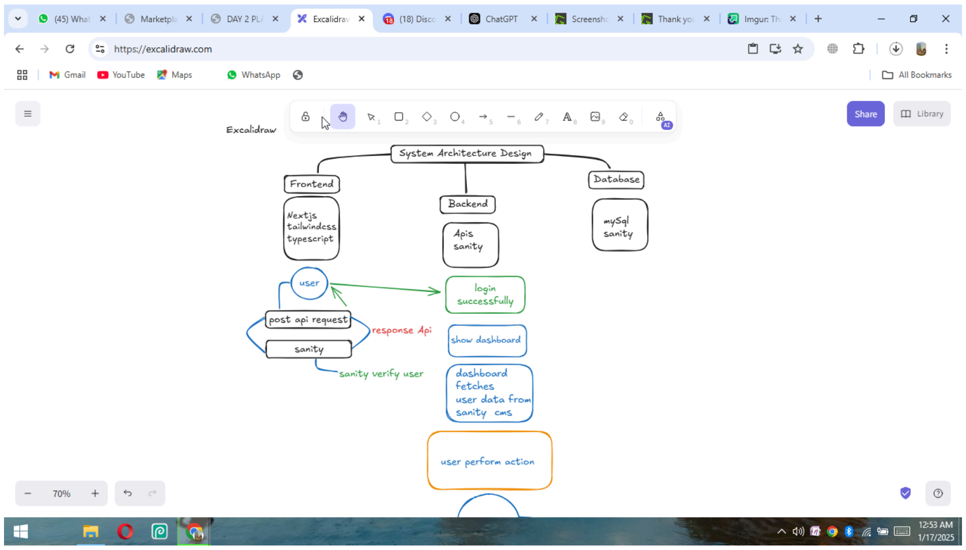
Backend: Updates Sanity CMS.

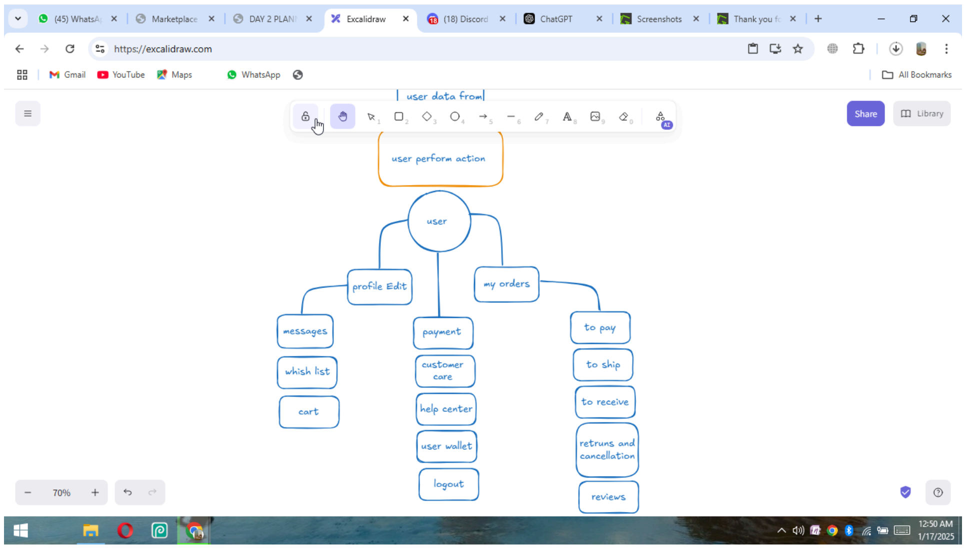
Frontend: Shows updated product list.

" Hafiza Ayesha Akbar "

Signature

RC


https://excalidraw.com
Excalidraw
System Architecture Design
Frontend
Nextjs
tailwindcss
typescript
Backend
Apis
sanity
Database
mySql
sanity
user
login successfully
post api request
response Api
sanity
sanity verify user
show dashboard
dashboard fetches user data from sanity cms
user perform action
70%
Share
Library
12:53 AM
1/17/2025


https://excalidraw.com
user data from
user perform action
user
profile Edit
my orders
messages
whish list
cart
payment
customer care
help center
user wallet
logout
to pay
to ship
to receive
retruns and cancellation
reviews
70%
Share
Library
12:50 AM
1/17/2025

Sanity Schema :

```
export default {
       name: 'product',
       type: 'document',
       title: 'Furniture Product',
       fields: [
       {
       name: 'name',
      type: 'string',
      title: 'Product Name'
      },
      {
        name: 'price',
       type: 'number',
       title: 'Price'
     },
   {

      name: 'stock',
      type: 'number',
      title: 'Stock Level'
     },
      {

     name: 'category',
     type: 'string',
      title: 'Category',



         options: {
                list: [
                 { title: 'Living Room', value: 'living_room' },
                 { title: 'Bedroom', value: 'bedroom' },
                { title: 'Office Furniture', value: 'office_furniture' }
     ]
  }
    },

   {
      name: 'image',
      type: 'image',
      title: 'Product Image',
      options: { hotspot: true }

    },

    {

      name: 'description',
       type: 'text',
       title: 'Description'
     },

    {

      name: 'dimensions',
       type: 'object',
       title: 'Dimensions',

      fields: [

        { name: 'width', type: 'number', title: 'Width (in cm)' },
        { name: 'height', type: 'number', title: 'Height (in cm)' },
        { name: 'depth', type: 'number', title: 'Depth (in cm)' }
    }]

};
```

Fields

1. name: Name of the furniture product.
2. price: Price of the product.
3. stock: Available stock quantity.
4. category: Predefined categories for organization (e.g., Living Room, Bedroom).
5. image: Image upload for the product with hotspot cropping.
6. description: Detailed description of the product.
7. dimensions: Dimensions of the furniture item (width, height, depth).