E-Commerce Website

Archit Sagar

Ashwin K

Overview:

Serve multiple users searching for products while also browsing and viewing products based on functionality like category and sorting. The Search/sort functionality is powered by the Unbxd API, while the Category functionality is powered by the API used to connect to the database.

Description:

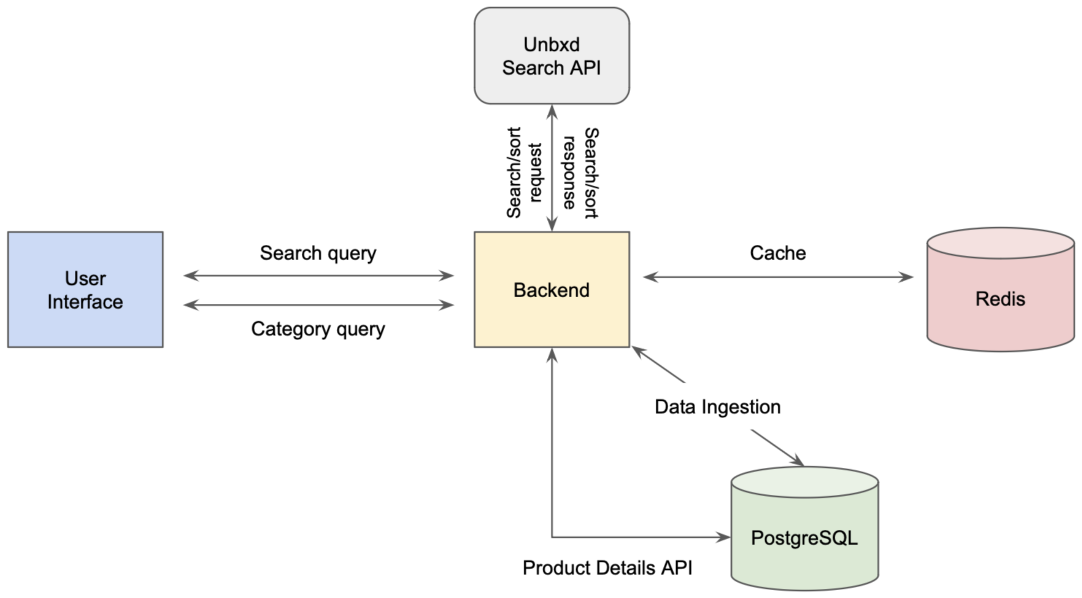
- Any user can view the products.
- Only one user role - Visitor.
  - Visitors can search for a product.
  - Visitors can view a particular product.
  - Optional - products can be recommended to users based on session actions.

Technologies:

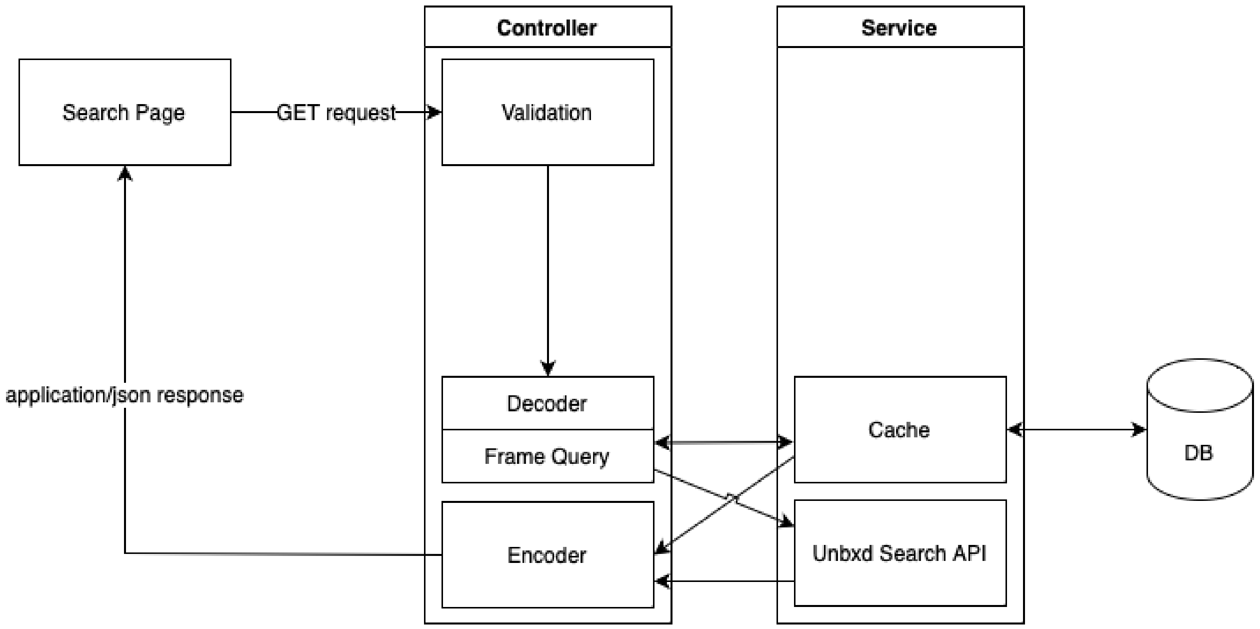
- Frontend - HTML, CSS, JavaScript
- Backend - Python-Flask
- Database -
  - Catalog information - PostgreSQL
  - Caches - Redis
  - Deployment - Kubernetes

Project Design:

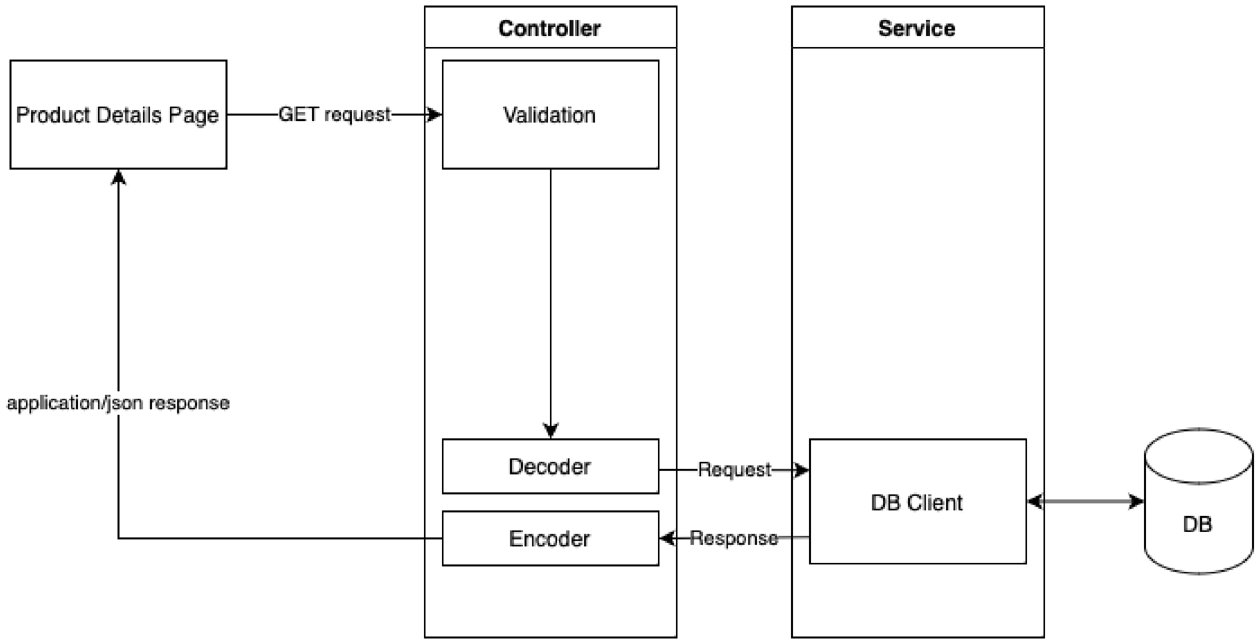
Application Diagram:



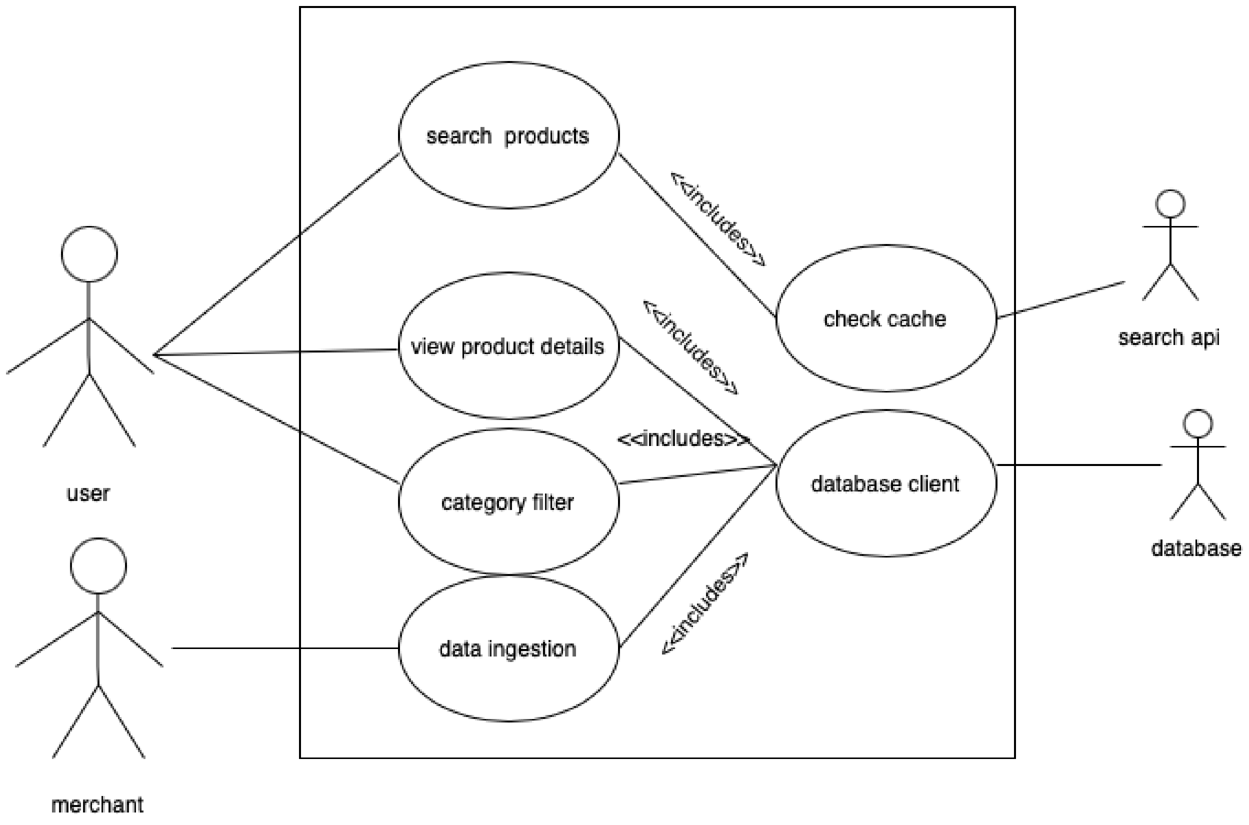
Search Page:



Product Details Page:



Use case diagram:



E-Commerce Website Details:

- Home page:
  - The first page which the user sees.
  - Displays plenty of featured products which the user might want to see.

- Search page:
  - Query typed in the search bar.
  - Based on the query, get the relevant products' information using the Unbxd API.
  - Display the products on the search page.
  - Maximum of 90 products are displayed, over 5 pages (i.e., 18 products per page).
  - The user can sort the products based on the price in the Ascending or Descending order.

- Category page:
  - 2 category levels are present
  - Given the selected category values, fetch the relevant products from the database.

- 18 products are displayed per page.
- Users can sort the displayed products based on the price in the Ascending or Descending order.

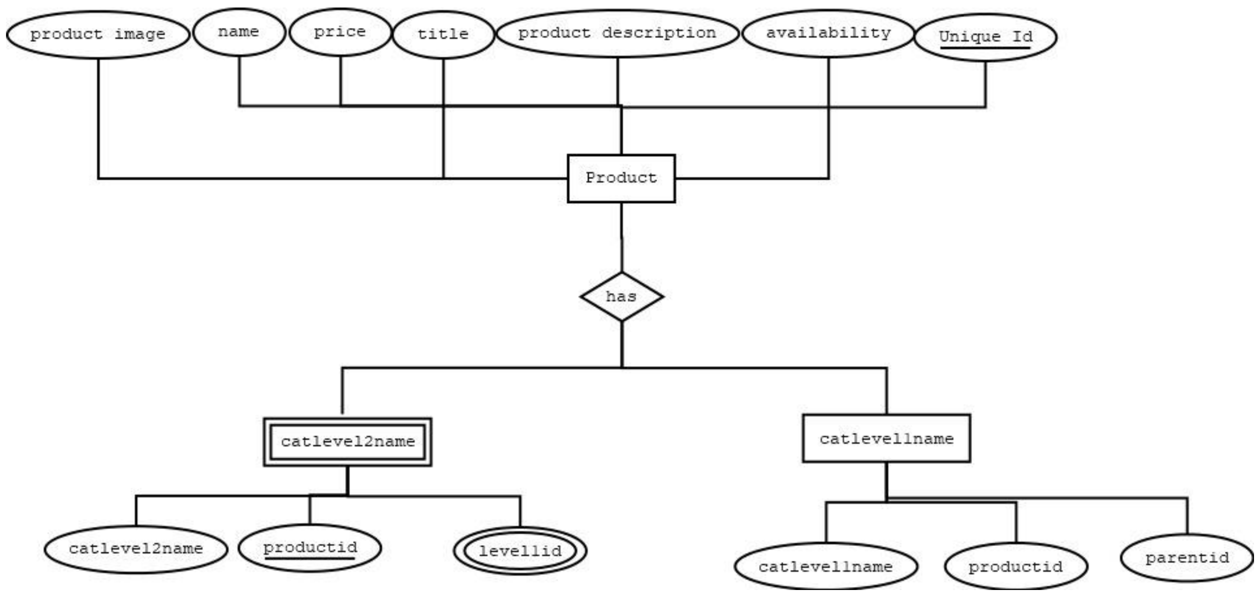
- Product details page:
  - Details of the selected product are displayed to the user.
  - Optional - retrieve relevant products based on user actions and display it at the bottom of the page.

Data Management

- Data Description:
  The database consists of:
  - Products: Information about all the products is stored with each product having a unique product ID.
  - Category: Information about the category levels and the products belonging to each level.

- Data Objects:
  - Products: ID, title, image, name, price, availability, description.
  - Category: ID, category, parent_id, productid, level.

- Database Relationships:



API Specifications:

Note: The API specifications were collected using the Postman testing tool.

1. **Ingestion:**
   - The ingestion API is used to insert the products from the products catalog into the database.
   - The ingestion API is also used to change the details of an existing product, i.e., update an existing product.
   - The ingestion API is exposed at `/ingestion`
   - The ingestion API takes 2 types of requests: POST(to insert products into the database) and PUT(to update an existing product).
   - API specs for POST request:

```
"name": "Ingestion insert",
           "request": {
               "method": "POST",
               "header": [],
               "body": {
                   "mode": "raw",
                   "raw": "<Data to be inserted>",
                      "options": {
                       "raw": {
                           "language": "json"
                       }
                   }
               },
               "url": {
                   "raw": "http://127.0.0.1:3000/ingestion",
                   "protocol": "http",
                   "host": [
                       "127",
                       "0",
                       "0",
                       "1"
                   ],
                   "port": "3000",
                   "path": [
                       "ingestion"
                   ]
               }
           },
           "response": []
       }
```

- API specs for PUT request:

```
"name": "Ingestion update",
           "request": {
               "method": "PUT",
               "header": [],
               "body": {
                   "mode": "raw",
                   "raw": "<Data to be updated>",
                      "options": {
                       "raw": {
                           "language": "json"
                       }
                   }
               },
               "url": {
                   "raw": "http://127.0.0.1:3000/ingestion",
                   "protocol": "http",
                   "host": [
                       "127",
                       "0",
                       "0",
                       "1"
                   ],
                   "port": "3000",
                   "path": [
                       "ingestion"
                   ]
               }
           },
           "response": []
```

2. **Header:**
   - The header API is used to get the required information for the header in each page of the website.
   - This API returns the information of the category levels.
   - The header API is exposed at `/`.
   - The header API gets only one type of request: GET request.
   - API specs for the GET request:

```
"name": "Header page",
           "request": {
               "method": "GET",
               "header": [],
               "url": {
                   "raw": "http://localhost:3000/",
                   "protocol": "http",
                   "host": [
                    "localhost"
                ],
                "port": "3000",
                "path": [
                    ""
                ]
            }
        },
        "response": []
```

3. **Home:**
    - The homepage API is used to get the required information for the home page.
    - The homepage API returns 18 products which will be displayed as featured products to the user.
    - The homepage API is exposed at `/home`.
    - The homepage API gets only one type of request: GET request.
    - API specs of the GET request:

```
"name": "Home page",
           "request": {
               "method": "GET",
               "header": [],
               "url": {
                   "raw": "http://localhost:3000/home/",
                   "protocol": "http",
                   "host": [
                       "localhost"
                   ],
                   "port": "3000",
                   "path": [
                       "home",
                       ""
                   ]
               }
           },
           "response": []
```

4. **Products:**
    - The products API is used to get the required information for the products page.
    - The products API returns all the products in the database, in the Ascending order.
    - The products API is exposed at `/products`.
    - The products API gets only one type of request: GET request.
    - API specs of the GET request:

```
"name": "Products page",
           "request": {
               "method": "GET",
               "header": [],
               "url": {
                   "raw": "http://localhost:3000/products?page=45",
                   "protocol": "http",
                   "host": [
                       "localhost"
                   ],
                   "port": "3000",
                   "path": [
                       "products"
                   ],
                   "query": [
                       {
                           "key": "page",
                           "value": "45"
                       }
                   ]
               }
           },
           "response": []
```

5. **Product-Details:**
   - The product-details API is used to get the required information for the product-details page.
   - The product-details API returns the information like product ID, title, price, description, image of the selected product.
   - The product-details API is exposed at `/products/<productID>` where productID is the unique ID of the product whose information is to be displayed.
   - The product-details API gets only one type of request: GET request.
   - API specs of the GET request:

```
            "name": "Product Details page",
            "request": {
                "method": "GET",
                "header": [],
                "url": {
                    "raw": "http://localhost:3000/products/04530243",
                    "protocol": "http",
                    "host": [
                        "localhost"
                    ],
                    "port": "3000",
                    "path": [
                        "products",
                        "04530243"
                    ]
                }
            },
            "response": []
```

6. Category:

- The category API is used to get the information for the category page.
- The category API returns information like category levels and details of the category of the lowest level.
- The category API is exposed at `/category`.
- The category API gets a parameter *order* which dictates the order in which the products are sent as response to the request.
- The category API gets only one type of request: GET request.
- API specs of the GET request:

```
"name": "Category asc",
           "request": {
               "method": "GET",
               "header": [],
               "url": {
                   "raw":
"http://localhost:3000/category?catlvl1=men&catlvl2=Tops&page=2&order
=Ascending",
                   "protocol": "http",
                   "host": [
                       "localhost"
                   ],
                   "port": "3000",
                   "path": [
                       "category"
                   ],
                   "query": [
                       {
                           "key": "catlvl1",
                           "value": "men"
                       },
                       {
                           "key": "catlvl2",
                           "value": "Tops"
                       },
                       {
                           "key": "page",
                           "value": "2"
                       },
                       {
                           "key": "order",
                           "value": "Ascending"
                       }
                   ]
               }
           },
           "response": []
```

7. **Search:**
    - The search API is used to get the required information for the search page.
    - The search API returns the details of the products which match the query which was given by the users.
    - The search API gets the query as one of the parameters and forwards it to the Unbxd API. The Unbxd API returns the products matching the query to the search API.
    - The search API can also get a parameter *order* which dictates the order in which the products are sent as response to the request.
    - The search API gets only one type of request: GET request.
    - API specs for the GET request:

```
"name": "Search desc",
           "request": {
               "method": "GET",
               "header": [],
               "url": {
                   "raw":
"http://localhost:3000/search?q=redspaceshirt&order=Descending&page=1
",
                   "protocol": "http",
                   "host": [
                       "localhost"
                   ],
                   "port": "3000",
                   "path": [
                       "search"
                   ],
                   "query": [
                       {
                           "key": "q",
                           "value": "redspaceshirt"
                       },
                       {
                           "key": "order",
                           "value": "Descending"
                       },
                       {
                           "key": "page",
                           "value": "1"
                       }
                   ]
               }
           },
           "response": []
```

Github Repository:
https://github.com/unbxdintership/ecommerce-website