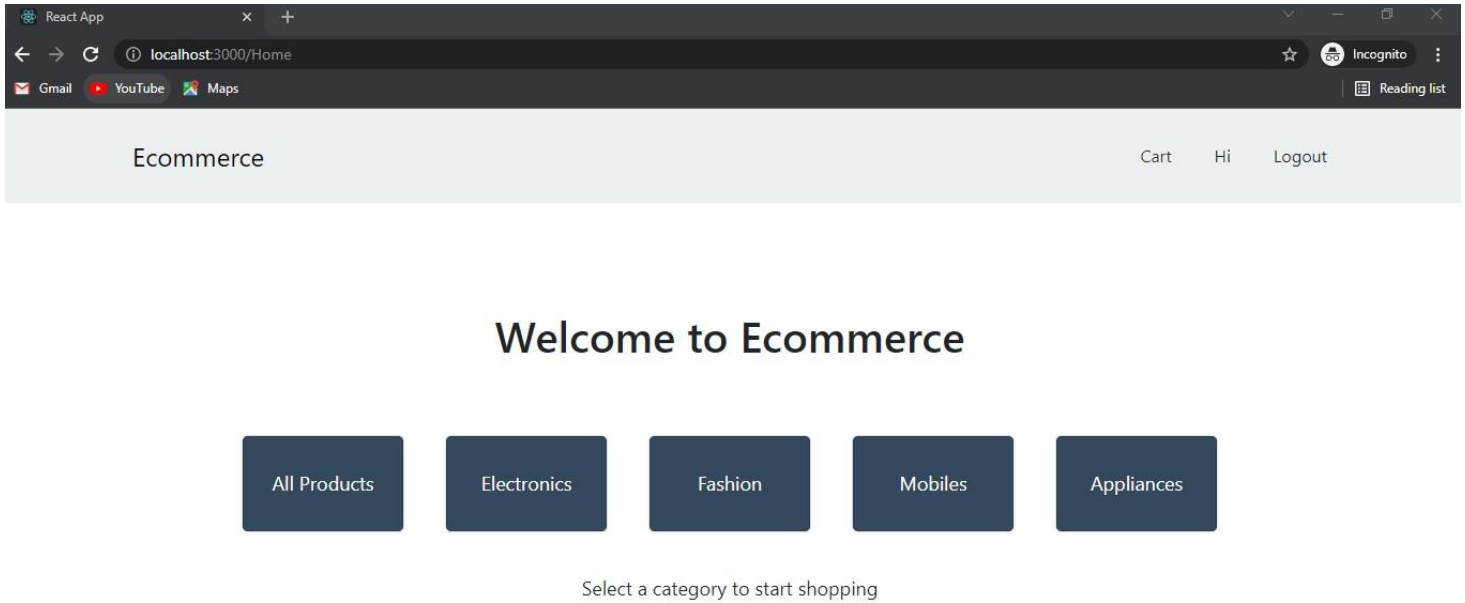
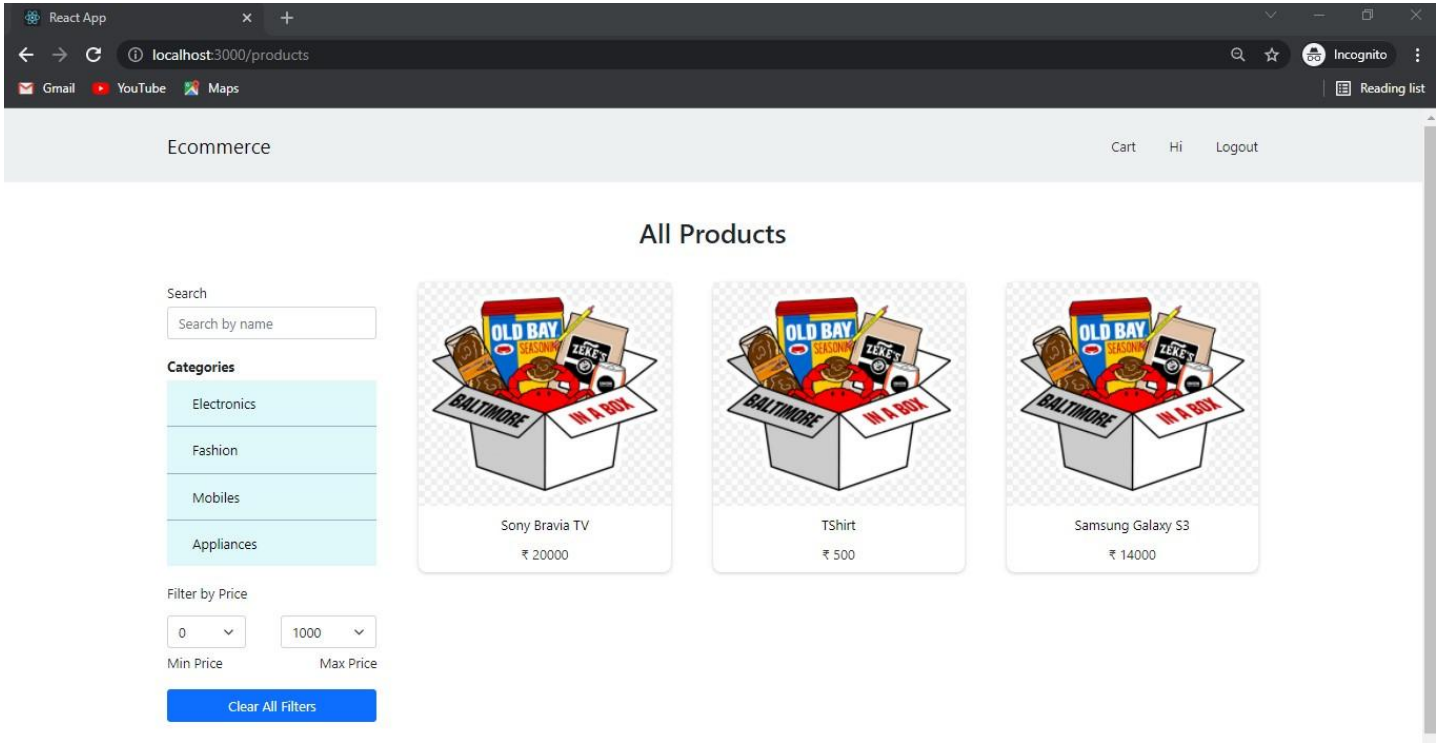
App Building: Ecommerce Website - Home Page and Product List Page

Relevel
by Unacademy

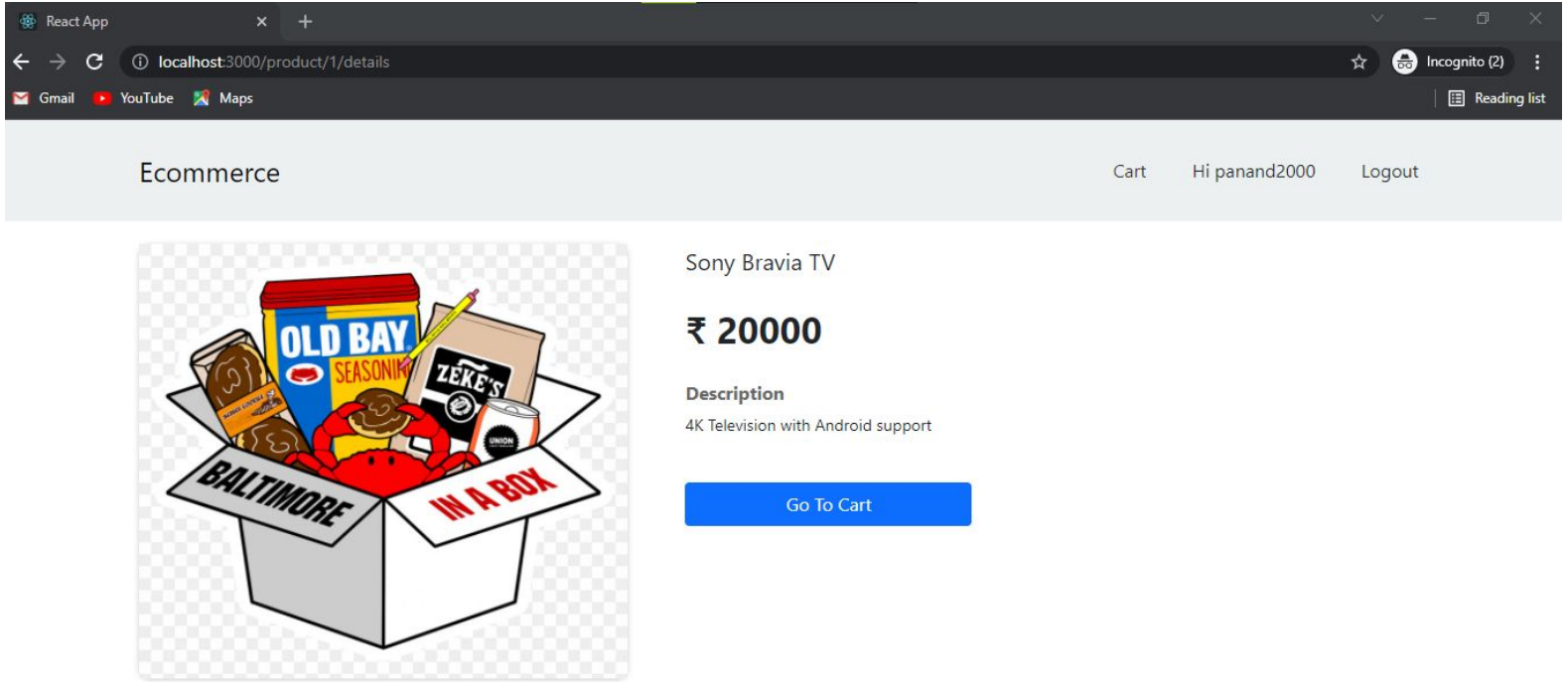
Home Page:


React App
localhost:3000/Home
Incognito
Gmail
YouTube
Maps
Reading list
Ecommerce
Cart
Hi
Logout
Welcome to Ecommerce
All Products
Electronics
Fashion
Mobiles
Appliances
Select a category to start shopping

Product List page:



ProductDetails page:



List Categories

Models

```
const sqlConnection = require("../services/sqlConnection");

module.exports = {
    listCategories: function(callback) {
        var sql = "SELECT ID as categoryId, Name as name FROM Categories";
        var values = [];
        sqlConnection.executeQuery(sql, values, function(err, result) {
            callback(err, result);
        });
    }
};
```

List Categories

Controller

```
const Category = require("../models/category");
const {
   httpCodes
} = require("../constants/backendConfig");

module.exports = {
   listCategories: function (req, res) {
       var responseData = {
           success: false,
           msg: "Error in fetching categories"
       };
       Category.listCategories(function (err, result) {
           if (err) {
               return res.status(httpCodes.internalServerError).send(responseData);
           }
           responseData.success = true;
           responseData.msg ="Successfully fetched categories";
           responseData.categories = result;
           return res.status(httpCodes.success).send(responseData);
       });
   }
};
```

List Categories

Index

```
extends layout

block content
 h1= title
 p Welcome to #{title}
```

Routes

```
var express = require('express');
var router = express.Router();

/* GET home page. */
router.get('/', function (req, res, next) {
   res.render('index', { title: 'Ecommerce' });
});

module.exports = router;
```

List Categories

Category Api Route

```
const express = require("express");
var router = express.Router();
const categoryController = require("../../../src/controllers/categoryController");



router.post("/category/all", categoryController.listCategories);
```

List Categories

Category Api Working

Request URL: http://localhost:3000/api/v1/category/all

Request Method: POST

Status Code: 200 OK

Remote Address: [::1]:3000

Referrer Policy: strict-origin-when-cross-origin


```
{success: true, msg: "Successfully fetched categories",…}
  categories: [{categoryId: 1, name: "Electronics"}, {categoryId: 2, name: "Fashion"},…]
    0: {categoryId: 1, name: "Electronics"}
      categoryId: 1
      name: "Electronics"
    1: {categoryId: 2, name: "Fashion"}
      categoryId: 2
      name: "Fashion"
    2: {categoryId: 3, name: "Mobiles"}
      categoryId: 3
      name: "Mobiles"
    3: {categoryId: 4, name: "Appliances"}
      categoryId: 4
      name: "Appliances"
    4: {categoryId: 5, name: "Groceries"}
      categoryId: 5
      name: "Groceries"
  msg: "Successfully fetched categories"
  success: true
```

Product Page

Filters

a. By name
b. By Category Id
c. By min Price
d. By max Price

List Products

Models

```
const sqlConnection = require("../services/sqlConnection");

module.exports = {
   listProducts: function(data, callback) {
          var sql = "SELECT ID AS productId, Name AS name, Price AS price FROM
Products";
      var values = [];
      if(data.categoryId) {
          sql += " WHERE CategoryId = ?"
          values.push(data.categoryId);
          if(data.query) {
              sql += " AND LOCATE('" + data.query + "', Name)";
          }
          if(data.minPrice) {
              sql += " AND Price >= " + parseInt(data.minPrice, 10);
          }
          if(data.maxPrice && parseInt(data.maxPrice, 10) > 0) {
              sql += " AND Price <= " + parseInt(data.maxPrice, 10);
          }
      } else if(data.query) {
          sql += " WHERE LOCATE('" + data.query + "', Name)";
          if(data.minPrice) {
              sql += " AND Price >= " + parseInt(data.minPrice, 10);
          }
          if(data.maxPrice && parseInt(data.maxPrice, 10) > 0) {
              sql += " AND Price <= " + parseInt(data.maxPrice, 10);
          }
      } else if (data.minPrice) {
          sql += " WHERE Price >= " + parseInt(data.minPrice, 10);
          if(data.maxPrice && parseInt(data.maxPrice, 10) > 0) {
              sql += " AND Price <= " + parseInt(data.maxPrice, 10);
          }
      } else if(data.maxPrice && parseInt(data.maxPrice, 10) > 0) {
          sql += " WHERE Price <= " + parseInt(data.maxPrice, 10);
      }

      sqlConnection.executeQuery(sql, values, function(err, result) {
          callback(err, result);
      });
   },
};
```

List Products

Controller

```
const Product = require("../models/product");
const {
   httpCodes
} = require("../constants/backendConfig");

module.exports = {
   listProducts: function (req, res) {
       var data = req.body;
       var responseData = {
           success: false,
           msg: "Invalid params for fetching products"
       };
       Product.listProducts(data, function (err, result) {
           if (err) {
               responseData.msg = "Error in fetching products";
               return res.status(httpCodes.internalServerError).send(responseData);
           }
           responseData.success = true;
           responseData.msg ="Successfully fetched products";
           responseData.products = result;
           return res.status(httpCodes.success).send(responseData);
       });
   }
};
```

List Products

Route

```
const express = require("express");
var router = express.Router();
const productController = require("../../../src/controllers/productController");

router.post("/product/all", productController.listProducts);

module.exports = router;
```

Add Products

Model

```
const sqlConnection = require("../services/sqlConnection");

module.exports = {
   addProduct: function(data, callback) {
          var sql = "INSERT INTO Products (Name, Price, Description, CategoryID,
VendorID, CreatedAt, UpdatedAt) "
           + "VALUES (?, ?, ?, ?, ?, now(), now())";
       var values = [];
       values.push(data.name);
       values.push(data.price);
       values.push(data.description);
       values.push(data.categoryId);
       values.push(data.vendorId);
       sqlConnection.executeQuery(sql, values, function(err, result) {
           callback(err, result);
       });
   },
};
```

Add Products

Controller

```
const Product = require("../models/product");
const {
   httpCodes
} = require("../constants/backendConfig");

module.exports = {

   addProduct: function (req, res) {
       var data = req.body;
       var responseData = {
           success: false,
           msg: "Invalid params for adding product"
       };
           if (data.name && data.price && data.description && data.categoryId &&
data.vendorId) {
           Product.addProduct(data, function (err) {
               if (err) {
                   responseData.msg = "Error in adding product";
                                                                                  return
res.status(httpCodes.internalServerError).send(responseData);
               }
               responseData.success = true;
               responseData.msg ="Successfully added product";
               return res.status(httpCodes.success).send(responseData);
           });
       } else {
           return res.status(httpCodes.badRequest).send(responseData);
       }
   },
};
```

Add Products

Route

```
const express = require("express");
var router = express.Router();
const productController = require("../../../src/controllers/productController");
router.post("/product/add", productController.addProduct);

module.exports = router;
```

Get Product Details

Model

```
const sqlConnection = require("../services/sqlConnection");

module.exports = {
   getProductDetails: function(data, callback) {
          var sql = "SELECT p.Name AS name, p.Price AS price, p.Description AS
description, if((SELECT COUNT(*) "
            + "FROM OrderDetails AS od LEFT JOIN OrderItems AS oi ON oi.OrderID =
od.ID WHERE oi.ProductID = p.ID"
           + "  AND od.UserID = ? AND od.OrderStatus = 1) > 0, 1, 0) AS addedToCart
FROM Products AS p WHERE p.ID = ? LIMIT 1";
       var values = [];
       values.push(data.userId);
       values.push(data.productId);
       sqlConnection.executeQuery(sql, values, function(err, result) {
           callback(err, result);
       });
   },
};
```

Get Products Details

Controller

```
const Product = require("../models/product");
const {
   httpCodes
} = require("../constants/backendConfig");

module.exports = {
   getProductDetails: function (req, res) {
       var data = req.body;
       var responseData = {
           success: false,
           msg: "Invalid params for fetching product details"
       };
       if (data.productId && data.userId) {
           Product.getProductDetails(data, function (err, result) {
               if (err) {
                   responseData.msg = "Error in fetching product details";
                                                                                   return
res.status(httpCodes.internalServerError).send(responseData);
               }
               responseData.success = true;
               responseData.msg ="Successfully fetched product details";
               responseData.productDetails = result[0];
               return res.status(httpCodes.success).send(responseData);
           });
       } else {
           return res.status(httpCodes.badRequest).send(responseData);
       }
   }
};
```

Get Products Details

Route

```
const express = require("express");
var router = express.Router();
const productController = require("../../../src/controllers/productController");

router.post("/product/details", productController.getProductDetails);

module.exports = router;
```

MCQs

1. 200, 400 status code stands for:
    a. success, bad request
    b. Bad request, success
    c. Internal error, Success
    d. Success, Internal error

MCQs

2. Modules can be exposed using:

a. expose
b. module
c. exports
d. All of the above

MCQs

3. HTTP status code for delete with NO CONTENT, when request body is empty?

    a. 200
    b. 201
    c. 204
    d. 304

MCQs

4. Which of the following is not an HTTP method?
    a. Create
    b. Put
    c. Post
    d. Option

MCQs

5. Which object is used to send the data to user.

a. Response Object
b. Request Object
c. Next Object
d. sqlConnection Object

Practice/HW

1. Create few products in the database using /product/add.
2. Add more filters in products.

Thank you