

UML Thrifty

Abstract:

The main objective of this project to design and develop a student-friendly responsive website for Thrift Store. We kept the website specific to students and is only used when logged in and authenticated by student email id. Also, this project was motivated by an urge to learn and implement the Progressive Web Application. The Website is built using ReactJS, html, CSS and Firebase for back-end programming.

Introduction:

Uml Thrifty web application is student specific application where user can register and login using their student email id and add their products to the thrift store by clicking on the button 'Add Products', by giving their contact information and uploading their product description and images. In the way all the students who have logged in can see the added products and contact the seller for the pick-up meetings. We have added dropdowns in order to filter the products.

We chose ReactJS for our web application because, it is no surprise that it is sweeping the codebase. A JavaScript library for front-end development that is open source is called ReactJS. Today's users demand faster, more dynamic websites, while programmers need a cutting-edge, adaptable development environment.

Web Technologies used:

Web technologies refers to the way computers/devices communicate with each other. We used front end and back end.

Front-End

Front-end refers to the web page's user interface. It is how the website appears to users when they visit it. The main front-end web technologies are described in the section that follows.

HTML (Hypertext Markup Language)

HTML describes the webpage structure. Tags are used to represent the many elements that make up HTML. These components specify how the content ought to be shown on the browser.

The screenshot shows a code editor interface with the following details:

- EXPLORER** tab is selected.
- PROJECT** sidebar shows files and folders:
 - public folder contains: logo192.png, logo512.png, manifest.json, robots.txt, umlthrift.ico, index.html (selected), logo.jpeg, Navbar.jsx, Product.jsx, Signup.css, Signup.jsx, Config.jsx, UserAuthContext.jsx, css folder (containing Home.css, Product.css), global folder (containing ProductsContext.jsx), App.jsx, App.test.js, and Dropdown.jsx.
 - src folder contains: AddProducts.jsx, Home.jsx, Login.jsx.
- index.html** file is open in the editor, showing its content:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="utf-8" />
  <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
  <meta name="viewport" content="width=device-width, initial-scale=1" />
  <meta name="theme-color" content="#000000" />
  <meta
    name="description"
    content="Web site created using create-react-app" />
  <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
  <!--
    manifest.json provides metadata used when your web app is installed on a
    user's mobile device or desktop. See https://developers.google.com/web/fundamentals
  -->
  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
  <!--
    Notice the use of %PUBLIC_URL% in the tags above.
    It will be replaced with the URL of the `public` folder during the build.
    Only files inside the `public` folder can be referenced from the HTML.
  -->
  <title>React App</title>
</head>
<body>
  <noscript>You need to enable JavaScript to run this app.</noscript>
  <div id="root"></div>
  <!--
    This HTML file is a template.
    If you open it directly in the browser, you will see an empty page.
  -->
</body>
</html>
```

CSS (Cascading Style Sheets)

CSS is used to style the HTML document. CSS can be defined inside an HTML file, injected into an HTML element, or applied externally as a (.css) extension.

The screenshot shows a code editor interface with a sidebar on the left displaying a project structure. The project structure includes a 'PROJECT' section at the top, followed by 'src', 'Components', 'Config', 'css', and several files like 'Login.jsx', 'Logo.jpeg', 'Navbar.jsx', etc. Below these are 'global', 'ProductsContext.jsx', 'App.jsx', 'App.test.js', 'Dropdown.jsx', 'index.css', 'index.jsx', 'reportWebVitals.js', 'setupTests.js', '.gitignore', 'package-lock.json', 'package.json', and 'README.md'. The 'css' folder contains 'Home.css' and 'Product.css', which is currently selected and highlighted in blue. The main editor area shows the CSS code for 'Product.css'.

```

src > css > # Product.css ...
1  @import url('https://fonts.googleapis.com/css?family=Open+Sans&display=swap');
2
3  /* products */
4
5  h1 {
6      padding: 30px;
7      border-bottom: 1px solid #e4e4e4;
8  }
9
10 @media(max-width: 768px) {
11     h1 {
12         text-align: center;
13         padding: 10px;
14     }
15 }
16
17 .products-container {
18     display: flex;
19     justify-content: center;
20     align-items: center;
21     padding: 20px;
22     flex-wrap: wrap;
23 }
24
25 .products-container .product-card {
26     width: 300px;
27     height: auto;
28     margin: 20px;
29     display: flex;
30     flex-direction: column;
31     justify-content: flex-start;
32     align-items: flex-start;
33     font-size: 12px;
34     font-weight: 600;
35     text-align: center;
36     box-shadow: 5px 5px 4px #e4e4e4;
37     position: relative;
38 }
39
40 .products-container .product-card .product-img {
41     width: 100%;
42     height: 200px;
43 }
44
45 .products-container .product-card .product-img img {

```

JavaScript

The most well-known object-oriented programming language for web pages is called JavaScript (JS). Programming on the client and server sides can both be done in JavaScript. JavaScript enables programmers to build animations, multimedia controls, and content that updates dynamically, among other

ReactJS

ReactJS, sometimes referred to as React or React.js, is a free JavaScript front-end development library. In order to create an interactive user interface, ReactJS is frequently utilized by developers. ReactJS is frequently preferred over its rival frameworks due to its simplicity, native approach, data binding, performance, and testability.

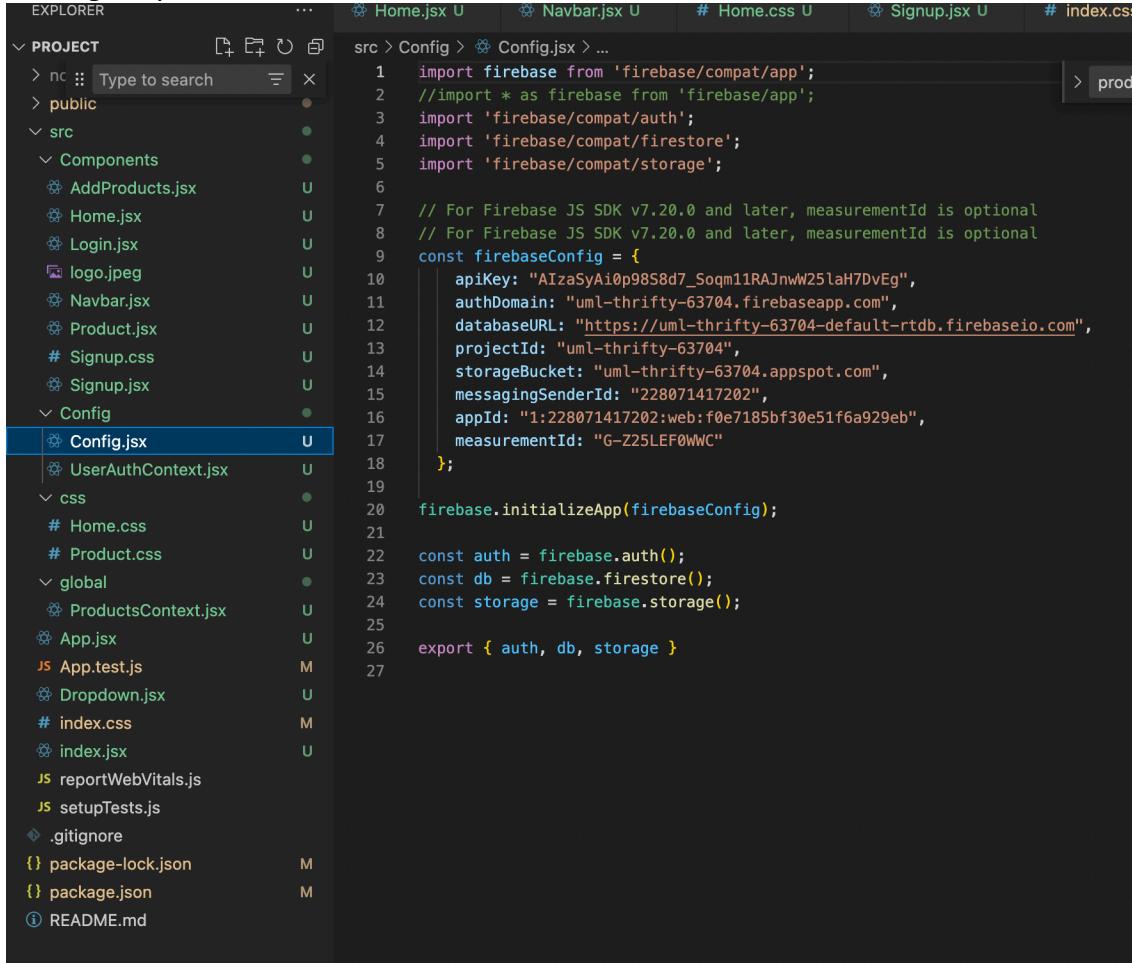
React Components

In React, components play a crucial role. Components are discrete, reusable User Interface elements that supply the display with data. With the help of ReactJS, it is possible to build smaller components, mix them, and nest them within one another to construct a whole user interface. The react elements that components, often referred to as props, return explain the contents of the user interface.

Back-end:

No SQL database for firebase:

Cloud Firestore is a NoSQL document database that lets you easily store, sync, and query data for your mobile and web apps - at global scale. It is a **Backend-as-a-Service (BaaS) app development platform** that provides hosted backend services such as a realtime database, cloud storage, authentication, crash reporting, machine learning, remote configuration, and hosting for your static files.



```
src > Config > Config.jsx > ...
1 import firebase from 'firebase/compat/app';
2 //import * as firebase from 'firebase/app';
3 import 'firebase/compat/auth';
4 import 'firebase/compat/firestore';
5 import 'firebase/compat/storage';
6
7 // For Firebase JS SDK v7.20.0 and later, measurementId is optional
8 // For Firebase JS SDK v7.20.0 and later, measurementId is optional
9 const firebaseConfig = {
10   apiKey: "AIzaSyAi0p98S8d7_Soqm11RAJnwW25laH7DvEg",
11   authDomain: "uml-thrifty-63704.firebaseioapp.com",
12   databaseURL: "https://uml-thrifty-63704-default-rtbd.firebaseio.com",
13   projectId: "uml-thrifty-63704",
14   storageBucket: "uml-thrifty-63704.appspot.com",
15   messagingSenderId: "228071417202",
16   appId: "1:228071417202:web:f0e7185bf30e51f6a929eb",
17   measurementId: "G-Z25LEF0WWC"
18 };
19
20 firebase.initializeApp(firebaseConfig);
21
22 const auth = firebase.auth();
23 const db = firebase.firestore();
24 const storage = firebase.storage();
25
26 export { auth, db, storage }
```

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure under **PROJECT**. The **src** folder contains **Components**, **Config**, and **css** subfolders, along with files like **AddProducts.jsx**, **Home.jsx**, **Login.jsx**, **logo.jpeg**, **Navbar.jsx**, **Product.jsx**, **Signup.css**, **Signup.jsx**, **UserAuthContext.jsx**, and **App.jsx**.
- CODE EDITOR**: Displays the **ProductsContext.jsx** file. The code is a class component that fetches products from a MongoDB collection and updates the state with changes. It uses the Context API to provide the product data to child components.

```
const prevProducts= this.state.products;
db.collection('Products').onSnapshot(snapshot=>{
  let changes= snapshot.docChanges();
  changes.forEach(change=>{
    if(change.type==='added'){
      prevProducts.push({
        ItemID:change.added.id,
        ItemName: change.added.data().ItemName,
        Category: change.added.data().Category,
        Location: change.added.data().Location,
        Description: change.added.data().Description,
        ProductPrice: change.added.data().ProductPrice,
        Phononenumber: change.added.data().Phononenumber,
        ProductImg: change.added.data().ProductImg,
        Status: change.added.data().Status
      })
    }
  })
  this.setState({
    products: prevProducts
  })
})
```

Implementation:

Project Structure:

Registration Page:

The website opens to be the sign-up page, and is only accessed after authenticated, user must make sure that they login only using their student email id.

UML THRIFTY

localhost:3000/signup

UML THRIFTY

Registration Form

Already have an account? [login](#)

Login Page:

Uml Thrifty is not a public page, and the content can be seen and modified only by the users who has logged in using their email id.

UML THRIFTY

localhost:3000/Home

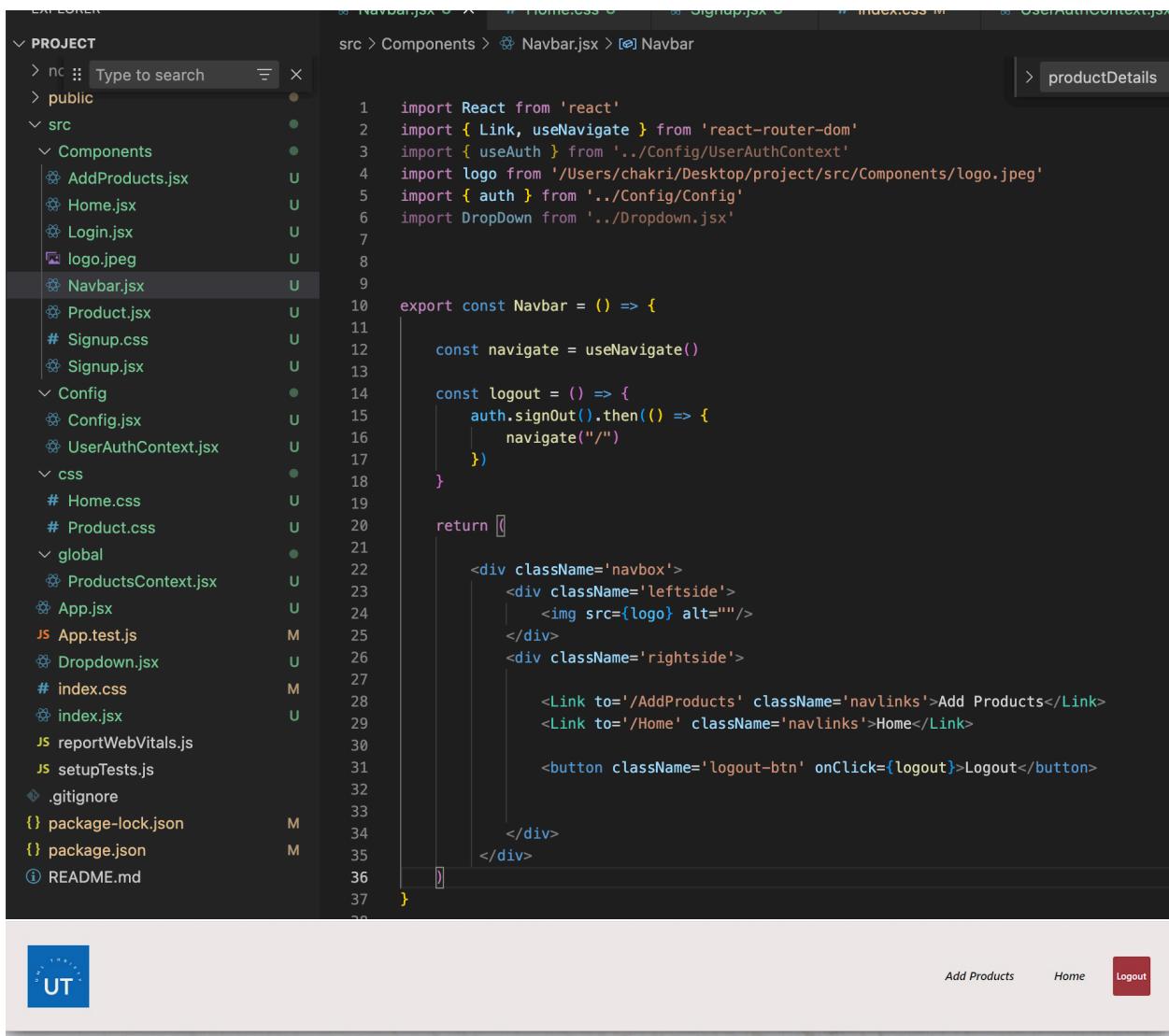


Add Products Home Logout

| | | | |
|---|--|--|---|
|  Computer Accessories mouse Gaming mouse with mouse pad \$ 15.00 Contact no: |  Automobile & Bicycles Bicycle 26' Shocker Mountain Bike \$ 90.00 Contact no: |  fadffh hkjbjk \$ 1000.00 Contact no: 9491605971 |  Automobile & Bicycles kbikfcwe ohj \$ 67.00 Contact no: 7 |
|  Sofa recently bought, not needed € 116.00 |  Automobile & Bicycles car Hyundai car |  gaming chair 6 months old chair with full functionality \$ 60.00 |  kjshih jhgukg \$ 100.00 |

Navigation bar:

Our navigation bar has home, add-products, and log-out options. Once, the user has logged in, the user will be redirected to the home page, and he/she can add their products by clicking on 'Add Products'.



```

  import React from 'react'
  import { Link, useNavigate } from 'react-router-dom'
  import { useAuth } from '../Config/UserAuthContext'
  import logo from '/Users/chakri/Desktop/project/src/Components/logo.jpeg'
  import { auth } from '../Config/Config'
  import DropDown from '../Dropdown.jsx'

  export const Navbar = () => {
    const navigate = useNavigate()

    const logout = () => {
      auth.signOut().then(() => {
        navigate("/")
      })
    }

    return [
      <div className='navbox'>
        <div className='leftside'>
          <img src={logo} alt="" />
        </div>
        <div className='rightside'>
          <Link to='/AddProducts' className='navlinks'>Add Products</Link>
          <Link to='/Home' className='navlinks'>Home</Link>

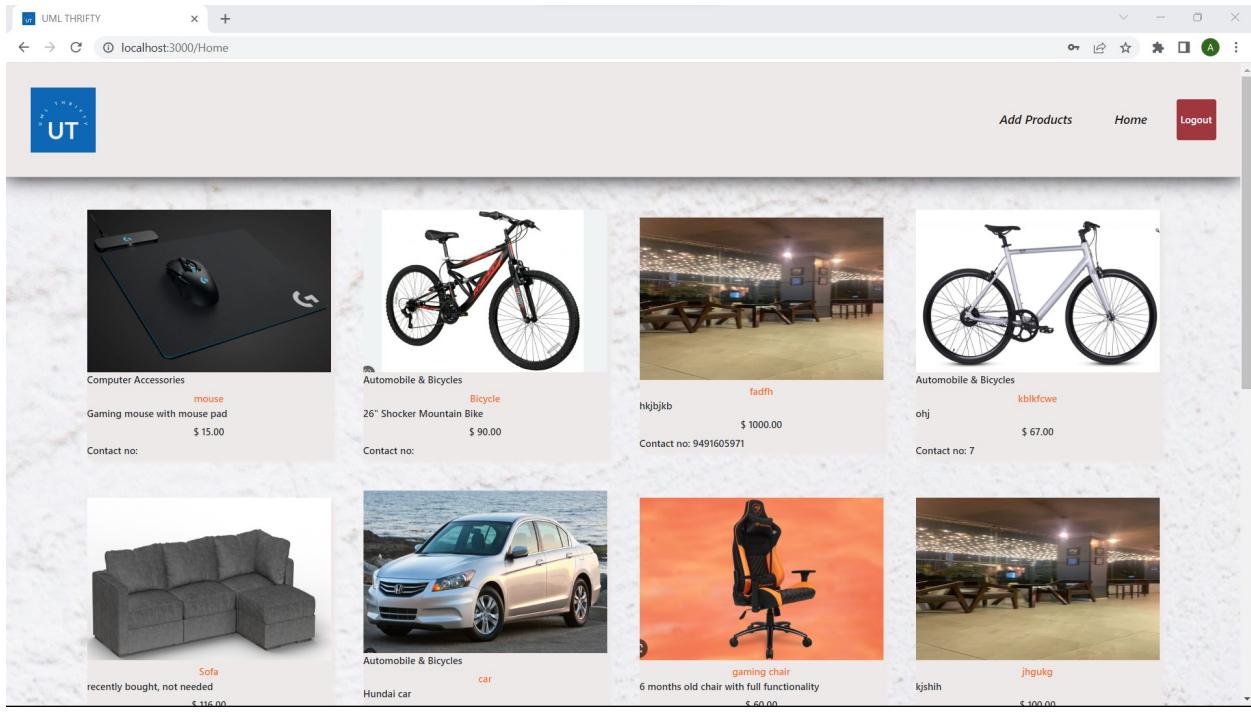
          <button className='logout-btn' onClick={logout}>Logout</button>
        </div>
      </div>
    ]
  }
}

```

Our navigation bar has home, add-products, and log-out options. Once, the user has logged in, the user will be redirected to the home page, and he/she can add their products by clicking on 'Add Products'.

Home Page:

Our home page has its navigation bar displayed on top and it has a filter using drop down by category of the products displayed. All the products can be displaying with their images and details mentioned like contact information, location, status of the product, cost of the product, title and description of the products.



Add Products:

On clicking Add Products from home page, a new page is opened with the navigation bar on top and has all its details like contact information, location, status of the product, cost of the product, title and description of the products to be entered and attach a .png or .jpg or .jpeg image file and should click on 'Add' button.

The screenshot shows a web browser window titled 'UML THRIFTY' with the URL 'localhost:3000/AddProducts'. The page has a header with a logo, 'Add Products', 'Home', and 'Logout' buttons. The main content is a form titled 'ADD PRODUCTS' with the following fields:

- Category**: Furniture
- Location**: Uml North
- Item**: (empty input field)
- Description**: (empty input field)
- Product Price**: 0
- Product Image**: Choose File (No file chosen)
- Contact Number**: (empty input field)
- Status**: (empty input field)

At the bottom of the form is a green 'ADD' button.

The screenshot shows the VS Code interface with the following details:

- Explorer View:** Shows the project structure. The `src` folder contains components like `AddProducts.jsx`, `Home.jsx`, `Login.jsx`, etc.
- File List:** The current file is `# index.css M`. Other files listed include `UserAuthContext.jsx U`, `ProductsContext.jsx U`, and `Product.jsx U`.
- Code Editor:** The code for `AddProducts.jsx` is displayed. It includes state management using `useState` for category, location, item, description, product price, product image, phone number, error, and status. It also includes a file selection handler for product images and a function to add a product to storage.

```
src > Components > AddProducts.jsx > AddProducts > addProduct > uploadTask.on('state_changed') c
  2 import {storage, db} from '../Config/Config'
  3 import { Navbar } from './Navbar';
  4 //import {Login} from './Login'
  5
  6
  7
  8 export const AddProducts = () => {
  9
 10     const [category, setCategory] = useState('');
 11     const [location, setLocation] = useState('');
 12     const [Item, setItem] = useState('');
 13     const [Description, setDescription] = useState('');
 14     const [ProductPrice, setProductPrice] = useState(0);
 15     const [ProductImg, setProductImg] = useState(null);
 16     const [Phonenumber, setPh] = useState('');
 17     const [error, setError] = useState('');
 18     const [Status, setStatus] = useState('');
 19
 20
 21
 22     const types = ['image/png', 'image/jpeg', 'image/jpg'];
 23
 24     const productImgHandler = (e) =>{
 25         let selectedFile = e.target.files[0];
 26         if(selectedFile && types.includes(selectedFile.type)) {
 27             setProductImg(selectedFile);
 28             setError('');
 29         }
 30         else{
 31             setProductImg(null);
 32             setError('Please select a valid image type');
 33         }
 34     }
 35
 36     const addProduct = (e) =>{
 37         e.preventDefault();
 38         //console.log(category, Item, Description, ProductPrice, ProductImg);
 39         const uploadTask = storage.ref(`product-images/${ProductImg.name}`).put(ProductImg);
 40         uploadTask.on('state_changed', snapshot=>{
 41             const progress =(snapshot.bytesTransferred/snapshot.totalBytes) * 100;
```

Firebase:

All the details of users entered are stored to firebase, an authentication link from firebase is sent to email id. When logged from the login page, it checks from the stored details of user. In the same way all the details of products are added to firebase, and retrieved from firebase to display on the home page.

The screenshot shows the Cloud Firestore Data page for a database named "uml-Thrifty". The left sidebar has a dark theme with icons for Home, Settings, and Firestore. The main area shows the "Products" collection under the "Products" subcollection. A document named "p8pWLW8Jv4U0Fx4rGUz5" is selected, displaying its details:

- Category:** "Home Appliances"
- Description:** "Scorpion Plus Hand Vacuum Cleaner"
- Item:** "Vacuum cleaner"
- ProductImg:** "https://firebasestorage.googleapis.com/v0/b/uml-thrifty-63704.appspot.com/o/product-images%2FScreen%20Shot%202022-12-06%20at%203.49.38%20PM.png?alt=media&token=13fd7bf5-ff20-4a50-8ea5-eac08823a677"
- ProductPrice:** 53

The right sidebar includes links to "Go docs", "More in Google Cloud", and a user profile icon.

The screenshot shows the Cloud Firestore Data page for the same database "uml-Thrifty". The left sidebar is identical. The main area shows the "userinfo" collection under the "userinfo" subcollection. A document named "17TbWydqh5W1dnbnNWpdpt4cZd0p1" is selected, displaying its details:

- FullName:** "Nagaditya"

The right sidebar includes links to "Go docs", "More in Google Cloud", and a user profile icon.

Future additions:

A search bar can be added to filter the display content on home page, and it can be extended for multiple universities so registration must have an addition of their university details.

Conclusion:

We have explored the usage of reactJS and developed the web application to meet the objective of the project to keep the content specific to University students and kept it user friendly.