Shopping Website

A
CS814
Course Project Report

Submitted by:
Divya Meena (202CS007)
Swati Singh (202CS031)

Submitted to:
Mahendra Pratap Singh

Department of Computer Science and Engineering
National Institute of Technology Karnataka
P.O. Srinivasnagar, Surathkal, Mangalore-575025
Karnataka, India
January 2021

2. Table of Contents

3. Introduction

Shopping has long been considered a recreational activity by many. Shopping online is no exception.

3.1. Goals

We have designed an e-shopping portal. The goal of this application is to develop a web based interface for online retailers. The system would be easy to use and hence make the shopping experience pleasant for the users. The goal of this application is

- To develop an easy to use web based interface where users can search for products, view a complete description of the products and add the products to their carts.
- The Seller can add products over a portal.

3.2. Scope

- A payment portal can be added to the current system.
- The users could subscribe for price alerts which would enable them to receive messages when prices for products fall below a particular level.
- Users can have multiple shipping and billing information saved. During checkout they can use the drag and drop feature to select shipping and billing information.

3.3. Hardware Specifications

- Operating system : Ubuntu(Version-20.04)
- Model name- Intel(R) Core(TM) i5-7200U CPU @ 2.50GHz
- Ram size- 8 GB
- Server : Node server 10.19.0
- Node Package Manager(NPM): 6.14.4
- Running node.js applications:

  Node.js Application can be hosted on Linux,Windows or any other O.S. And for node.js Application there is a basic minimum setup is required. like Node.js , git bash, npm etc.

3.4. Technologies Used

- HTML
- CSS
- JavaScript
- NodeJs
- MongoDB Atlas
- Visual Studio (Editor)

3.4.1. Node.js

Node.js is a JavaScript environment provider. It allows you to execute JavaScript code on the server. Node.js is dependent on the Google V8 engine which is the core of Chrome browser. C and C++ run both Node and V8 which is better in performance speed and memory consumption.

A Node app runs in an individual process with event looping, without the need of a new thread for each operation. This is opposed to traditional servers which make use of limited thread to handle requests. Node is a non-blocking, asynchronous and event-driven engine aimed for scalable application development. Generally, Node libraries are created using a non-blocking pattern. Application calls a request and then moves on to work on the next task rather than stalling while waiting for a response. When the request is completed, a callback function informs the application about the results. This allows multiple connections or requests to a server to be executed simultaneously which is important when scaling applications. MongoDB is also invented to be used asynchronously, therefore it is compatible with Node.js applications.

3.4.2. MongoDB Atlas

MongoDB Atlas is the cloud service for storing data globally. It offers everything MongoDB provides without further requirements when building applications, allowing developers to fully focus on what they do best. With Atlas, developers do not have to worry about paying for unnecessary things as they follow a pay-as-you-go standard. MongoDB Atlas introduces a simple interface to get started and running. Basically, by choosing instance size, region and any customized features needed, Atlas brings out the suitable instance and includes it with the following:

- Built-in Replication: Atlas ensures constant data availability by providing multiple servers, even when the primary server is down.
- Backups and Point-in-time Recovery: Atlas puts a lot of efforts to prevent data corruption.
- Automated Update and One-Click Upgrade: users take advantage of the latest and best features as soon as they are released due to Atlas automated patching.
- Various options for additional tools: users can freely decide on which regions, cloud providers and billing options they prefer to use, making them feel like customizing their own instances.

3.4.3. Node Package Manager (NPM)

NPM offers two essential functionalities:

- Command line for performing NPM commands such as package installation and version management of Node packages
- Downloadable repositories for Node package.

Beginning with being a method to download and manage Node dependencies, npm has become a powerful means in front-end JavaScript. There are two types of Node packages

installation, globally or locally by the npm install command. The package then is downloaded from the NPM registry and presented in a folder named node_modules. The package is also added to the package.json file under property dependencies.

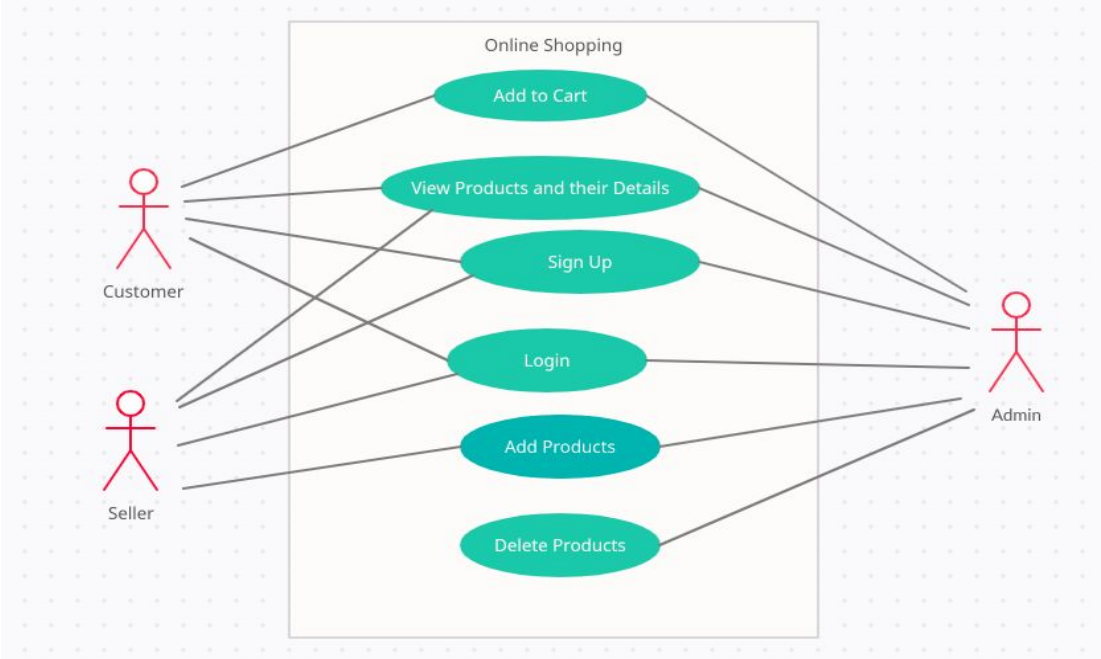
3.5. Implementation Details

Our project implements a part of e-shopping platform. It reflects the relationship of three entities -'Consumer', 'Seller' and 'Admin'.

We are trying to implement the following functionalities:

A normal user needs to sign up in order to view the products of the page.
A 'Customer' can add products in the shopping carts which are currently present on site.
A 'Seller' can add their products over the portal.
'Admin' canada or delete the 'Product' as per the requirement. Admin will act as a supreme authority on this portal. Admin cannot surf through the carts of the customer. Admin cannot alter the details of the Customer or Seller.

3.6 Use Case Diagram:



4.Authorization

4.1 Need of RBAC Authorization

Role-based access control (RBAC) refers to the idea of assigning permissions to users based on their role within an organization. It offers a simple, manageable approach to access management that is less prone to error than assigning permissions to users individually.

When using RBAC we can assign one or more roles to each user and one or more permissions to each role. The user-role and role-permissions relationships make it simple to perform user assignments since users no longer need to be managed individually, but instead have privileges that conform to the permissions assigned to their role(s).

In our model, higher components inherit the permissions of lower components.
A user can have more than one role which makes it crucial to implement the RBAC model.

4.2. $RBAC_1$ Model Definition

In our Project, the $RBAC_1$ model is adopted from *Ravi S. Sindhu [1]* research paper.

1. U, R, P , and S (users, roles, permissions and sessions respectively)
2. PA ⊆ P x R, a many-to-many permission to role assignment relation,
3. UA ⊆ U x R, a many-to-many user to role assignment relation,
4. user : S➡U, a function mapping each session s_i to the single user user(s_i) (constant for the session's lifetime)
5. RH R xR is a partial order on R called the role hierarchy or role dominance relation, also written as ≥, and
6. roles : S➡2^R , is modified from $RBAC_0$ to require roles(s_i) ⊆ {r | (∃ r´ ≥ r) │ (user (s_i),r´) ∈ UA │ } (which can change with time) and session s_i has the permissions $\bigsqcup_{r \in roles(s_i)}$ { p | (∃ r´´ ≥ r) │ (p, r´´) ∈ PA │ }.

4.3. Components of RBAC present in Application

We have implemented RBAC in our application using node.js:

- Setting up database with MongoDB
- Setting up User Authentication
    - Signup
    - Login

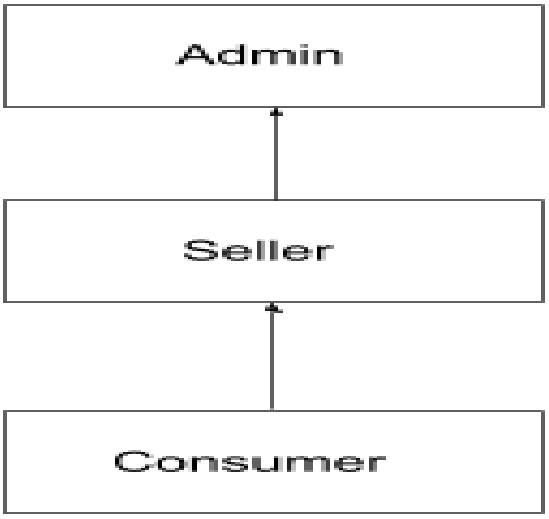
- Creating User Roles
- Setting up routes

We have implemented following Roles in our application:

1. Admin
2. Seller
3. Customer

When a user logs in successfully to your application, the authorization server send tokens to the client:

➔ *Access Token*
After a user successfully authenticates and authorizes access, the client application receives an access token from the authentication server. The client passes the access token as a credential whenever it calls a protected endpoint. This token informs the server that the client is authorized to access the application. Through its permissions claim, the access token tells the server which actions the client can perform on which resources.



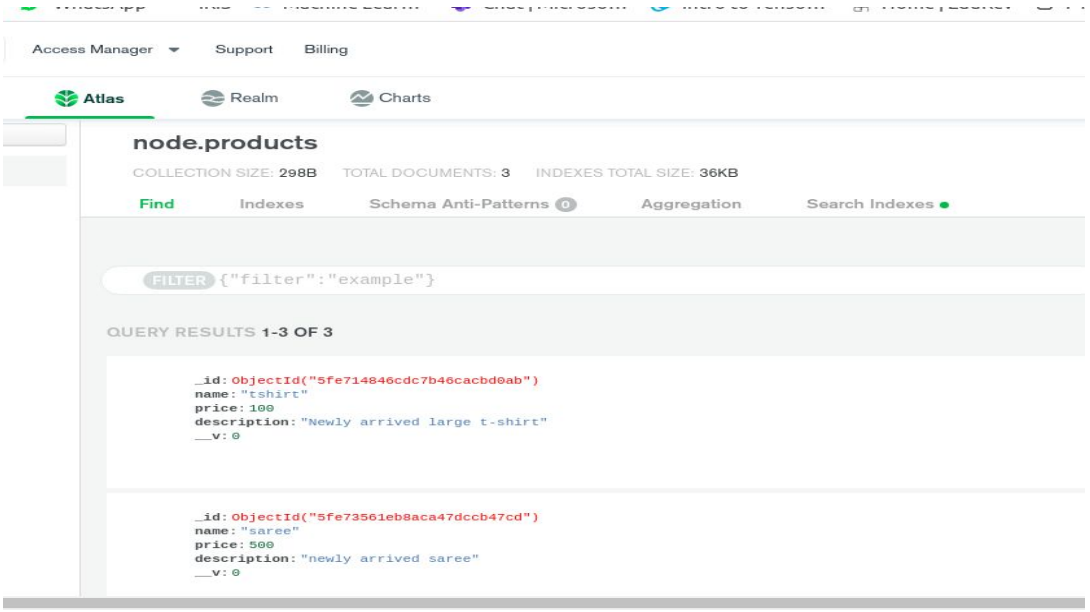
We are implementing the $RBAC_1$ model with three hierarchy levels -Customer, Seller and Admin.

Customer's permissions are inherited to Seller and Admin.

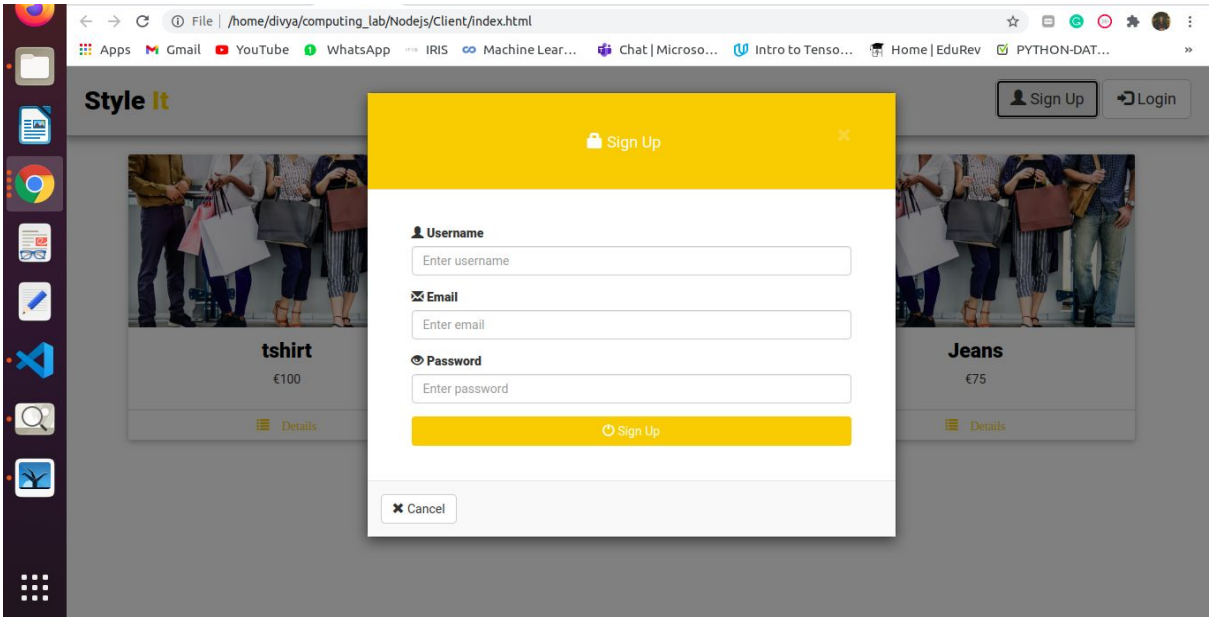
➔ All normal users are "Customers".
➔ An "Admin" can also be a "Customer" or "Seller".
➔ A "Seller" can also be a "Customer".
➔ A "Customer" can only purchase things available on cart.

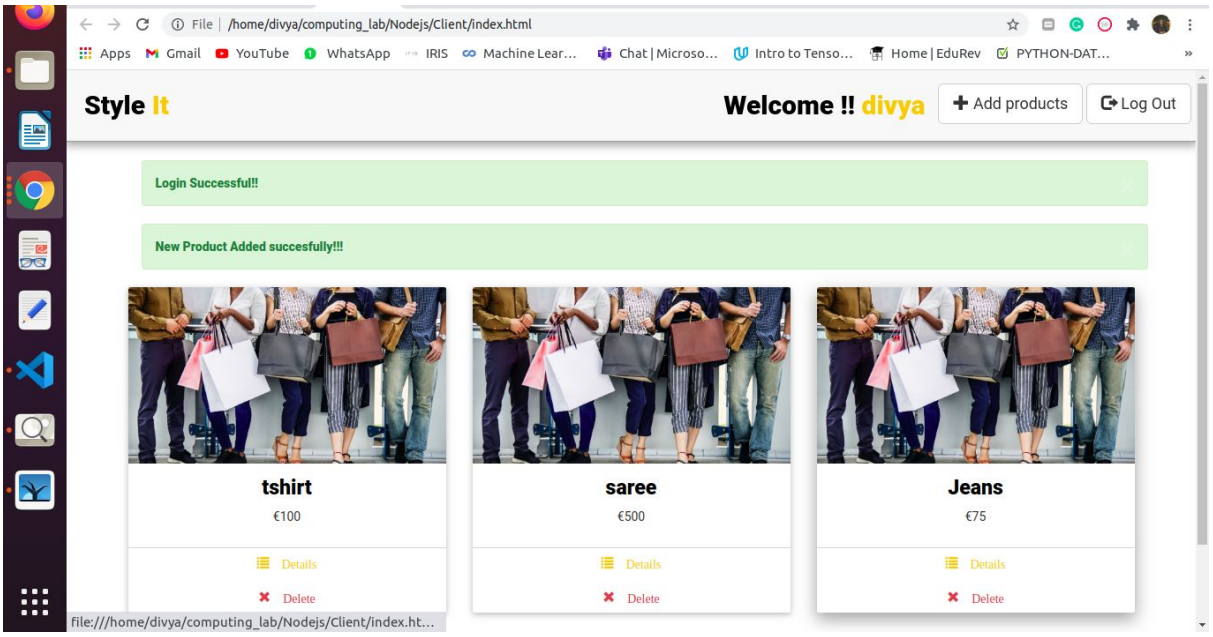
5. Snapshots

Structure of products table in database:



Login page and others :




File | /home/divya/computing_lab/Nodejs/Client/index.html
Apps
Gmail
YouTube
WhatsApp
IRIS
Machine Lear...
Chat | Microso...
Intro to Tenso...
Home | EduRev
PYTHON-DAT...
Style It
Welcome !! divya
Add products
Log Out
Login Successful!!
New Product Added succesfully!!!
tshirt
€100
Details
Delete
saree
€500
Details
Delete
Jeans
€75
Details
Delete
file:///home/divya/computing_lab/Nodejs/Client/index.ht...

6. Conclusion

The application can be used for any Online Shopping application. It is easy to use, since it uses the GUI provided in the user dialog. User friendly screens are provided. The 'Online Shopping' is designed to provide a web based application that would make searching, viewing and selection of a product easier. The user can then view the complete details of each product. RBAC1 is implemented in this application which introduces role hierarchy with the roles created in our application. In RBAC one user can perform more than one rule, and we implemented this advantage of RBAC by inheriting the roles. Admin can act as both seller and customer in our application and a seller can also act as customer. We make our website secure with RBAC implementation but it can be improved more.

Limitations: Users cannot save the shopping carts so that they can access later i.e. they cannot create wish lists which they can access later. This application does not have features by which users can set price ranges for products and receive alerts once the price reaches the particular range.

7. References

1. IEEE Computer, February 1996 Role-Based Access Control Models by Ravi S. Sandhu. https://csrc.nist.gov/CSRC/media/Projects/Role-Based-Access-Control/documents/sandhu96.pdf
2. A paper “Managing Multi-dimensional Multi-granular Security policies using Data Warehousing” [online] Available at: https://www.sciencedirect.com/science/article/pii/S0167404819301166
3. [online] Available at: https://medium.com/bluecore-engineering/implementing-role-based-security-in-a-web-app-89b66d1410e
4. [online] Available at: https://soshace.com/implementing-role-based-access-control-in-a-node-js-application/
5. [online] w3school. Node.js Introduction. Available at: https://www.w3schools.com/nodejs/nodejs_intro.asp
6. Tutorialspoint. Node.js - NPM. Available at: https://www.tutorialspoint.com/nodejs/nodejs_npm.htm
7. MongoDB inc. NoSQL Databases Explained. Available at: https://www.mongodb.com/nosql-explained