Sporty Shoes

Developed by Mohd Kaif.

Shaikhkaif3120@gmail.com

Sprints: -

Learner’s Academy was divided in 3 sprints of 1 week each.

Sprint 1: -

* Initialized a spring boot jar application with dependencies web, Jpa and thymeleaf
* Made E.R. diagram for database and entities relationship.
* Made classes for entities and established relationship between them.
* Configured hibernate and mapped classes as entities.
* Made repository and service layers
* Made front end pages for admin home, adding category, managing category, adding products, managing products

Sprint 2: -

* Planned the structure of the website and navigation across it.
* Made crud operations of categories and products.
* Made Admin controller and mapped the routes for crud operations related to product and categories
* Added bootstrap to the project to do styling
* Made User model with Roles model where user can have either user access or Admin access
* Made password changing page for admin
* Made shop page
* Made cart page and checkout page and set the routings for it

Sprint 3: -

* Added spring security to the project
* Set up admin routes just for the user with admin Authority
* Set up permit all links for which you don’t have to be logged in such as shop, home, login and register
* Made register page for adding users and configured login page
* Made Purchase model and it adds all the product to the checkouts to the purchase
* Made purchase reports accessible to only admin users
* Made filtered reports on the basis of category, date or both of them
* Made front end for all the functionalities

**Java Classes: -**

Spring boot main class -> SportyShoesApplication.java

Configuration folder: -

* GoogleOAuth2SuccessHandler
* MySecurityConfig

Controller folder: -

* AdminController
* HomeController
* CartController
* UserController
* PurchaseController

dto folder: -

* ProductDto

Global folder: -

* GlobalData

Model folder: -

* Category
* CustomUserDetails
* User
* Product
* Purchase
* Role

Repository folder: -

* UserRepository
* ProductRepository
* CategoryRepository
* PurchaseRepository
* RoleRepository

Service folder: -

* CategoryService
* CustomUserDetailService
* ProductService
* PurchaseService

**Resources Folder: -**

Static Folder: -

* Css folder
* Js folder
* Images folder
* Product images folder

Templates folder: -

* adminHome.html
* cart.html
* categories.html
* categoriesAdd.html
* changePassword.html
* checkout.html
* index.html
* login.html
* orderplaced.html
* products.html
* productsAdd.html
* purchase-report.html
* register.html
* shop.html
* viewProduct.html

**Routings: -**

Admin controller: -

* /admin -> for admin home
* /admin/products -> products list with update and delete option and add product option
* /admin/products/add (get)-> gets form for adding product
* /admin/products/add (post) -> adds the product
* /admin/products/delete/id-> deletes the product with the id
* /admin/products/update/id (get)-> gets the product form with data of that product so that you can alter and add the data of that product
* /admin/categories -> category list with delete and update options and an add category option
* /admin/categories/add (get) -> gets form for adding category
* /admin/categories/add(post) -> adds a category
* /admin/categories/delete/id -> deletes the category with that id
* /admin/categories/update/id -> gets the form with data of the category that will be updated
* /admin/changePassword (get) -> gets the form changing password
* /admin/changePassword (post) -> changes the password if you give existing password and it validates to be true

Purchase controller: -

* /admin/purchaseReports(get)-> gets the list of all the purchases with product name, category, picture and user email and date of purchase
* /admin/purchasedReports(post) -> gets the report on the basis of filter of categories or date or both of them at once
* /purchased -> saves the product and user that bought the product after checkout

LoginController: -

* /login(get) -> gets the login form
* /login (post)-> logins in the valid user (implementation by spring security)
* /register (get) -> gets the registration form
* /register (post) - > registers the user with the application
* /oauth2/authorization/google -> oAuth2 login by google accounts (Implementation by google and spring security)

HomeController: -

* /, /home -> gets the home page (index.html)
* /shop -> gets the shop page
* /shop/category/id -> gets the shop page based on category
* /shop/viewProduct/id -> shows the product with that product id

CartController: -

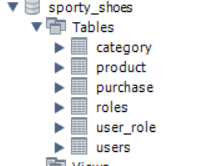
* /cart -> lists the product that user has added to cart
* /cart/removeItem/index -> removes the item from that list
* /addToCart/id -> adds the product with that id to the cart static list
* /checkout -> checkouts the product in the cart and add it to purchases

Core concepts, tools and technologies

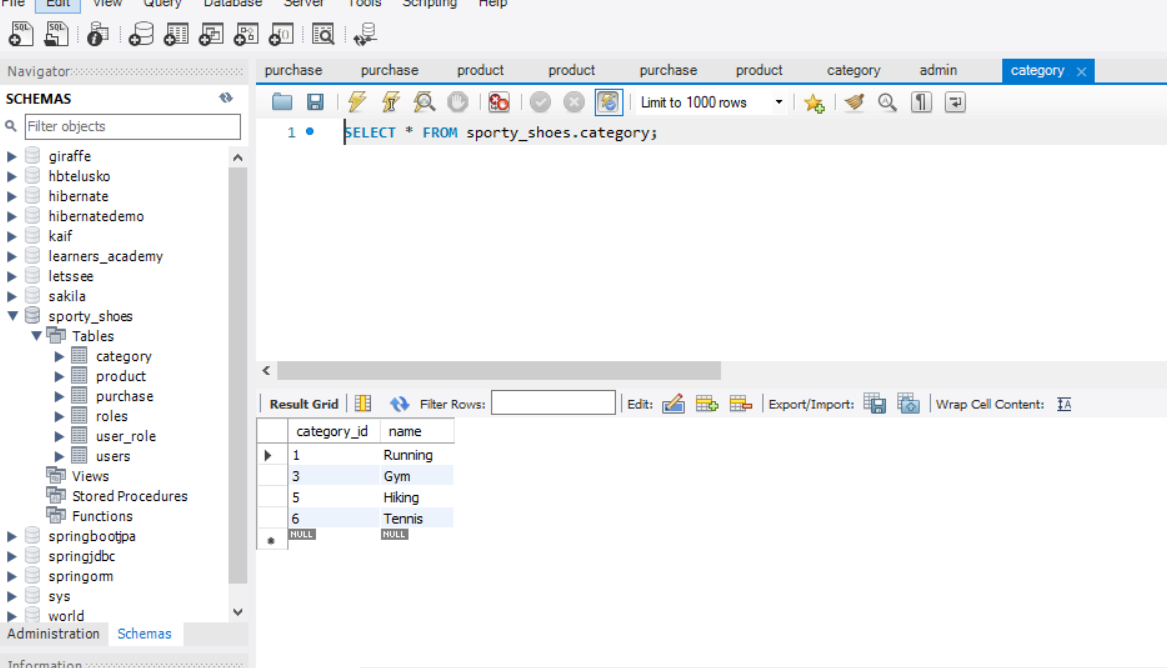
The core concepts that have been used in this project are hibernate (jpa) for entity management. Spring security for managing the user and sessions and web for controller for mapping routes. Thymeleaf as templating and view engine. Bootstrap for frontend designing.

The concepts such as controller and Java utility classes have played a major part in building of this project

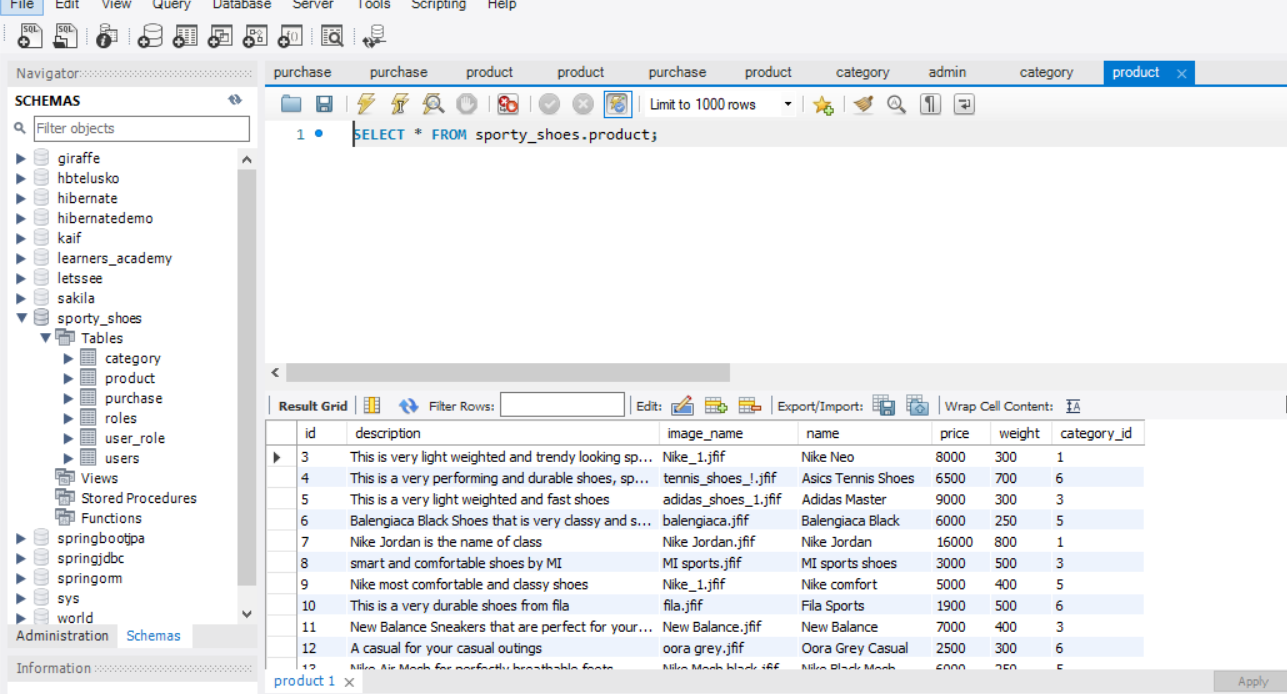
**Database Entity: -**



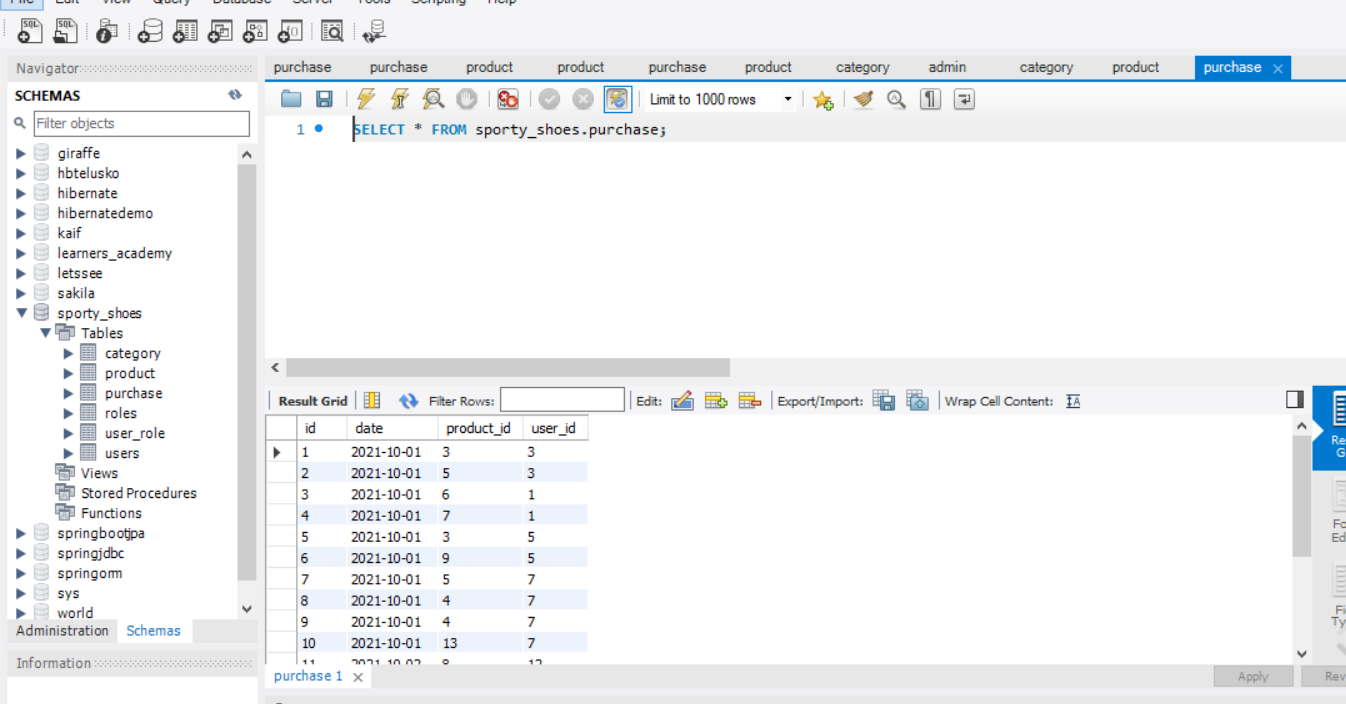
Category Entity



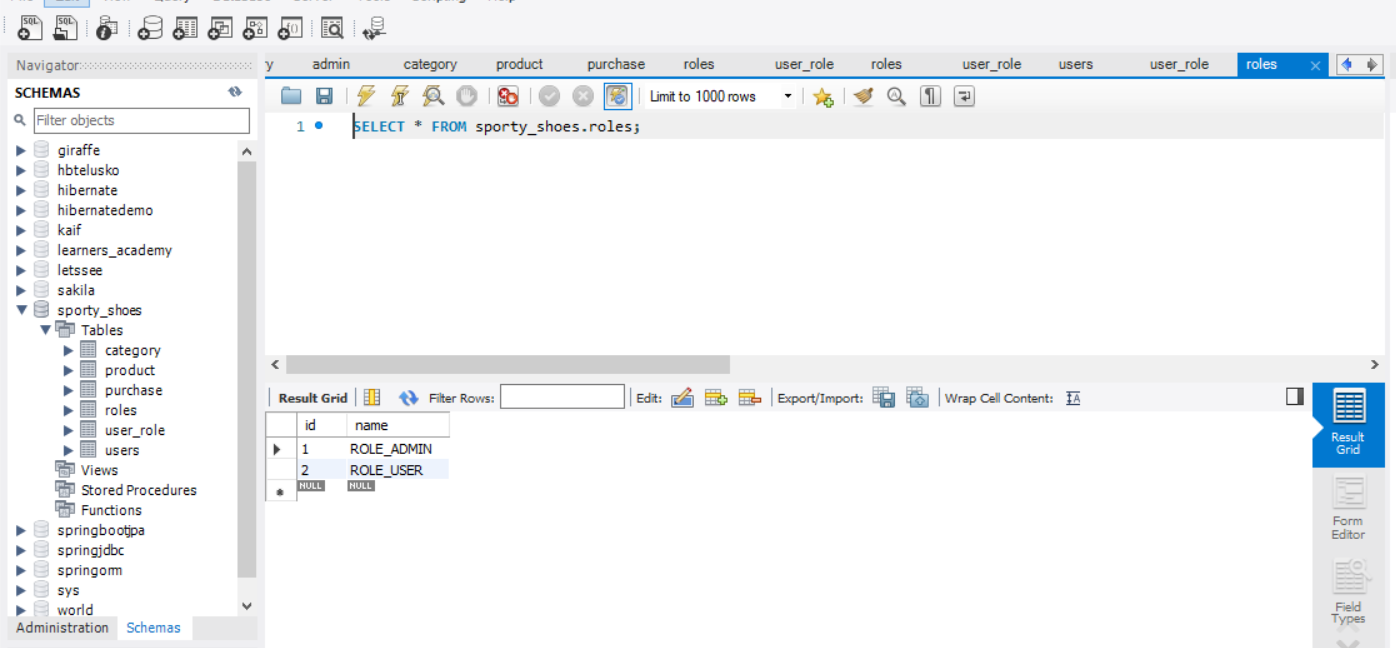
Product entity



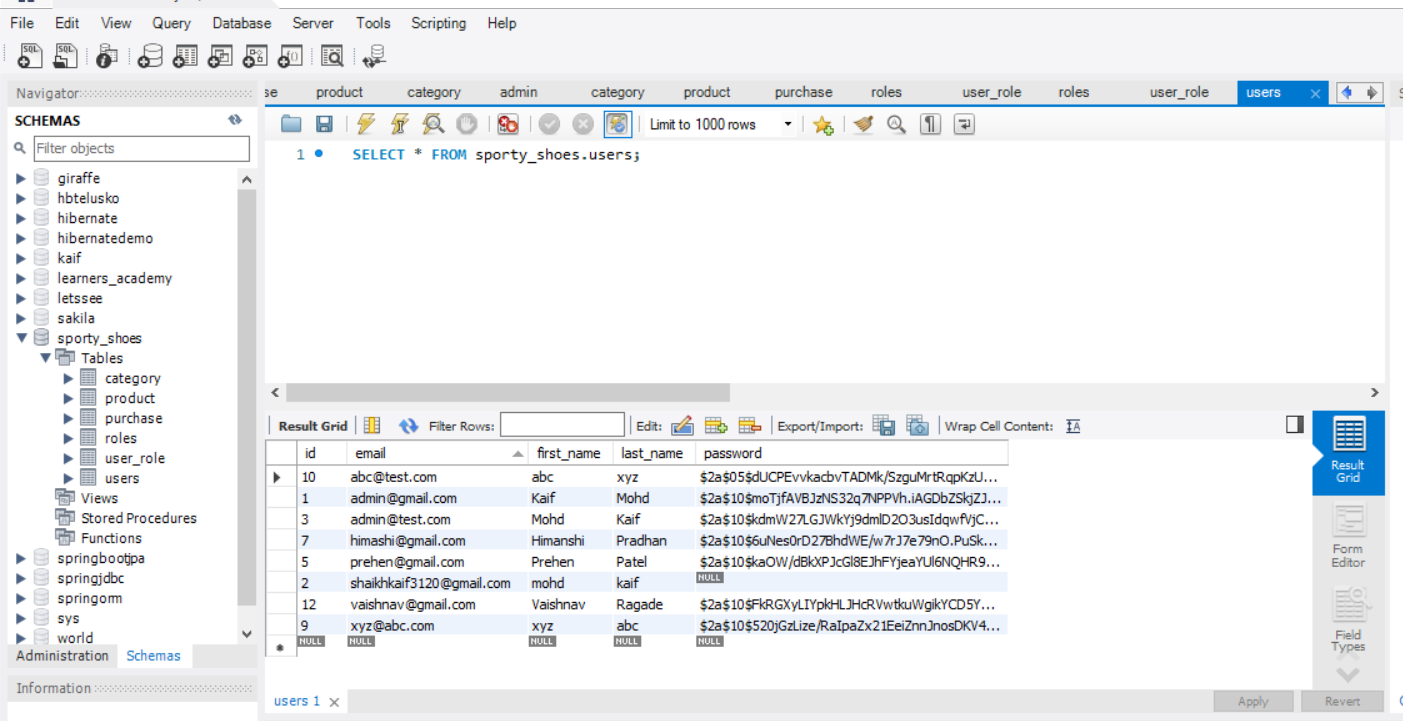
Purchase entity



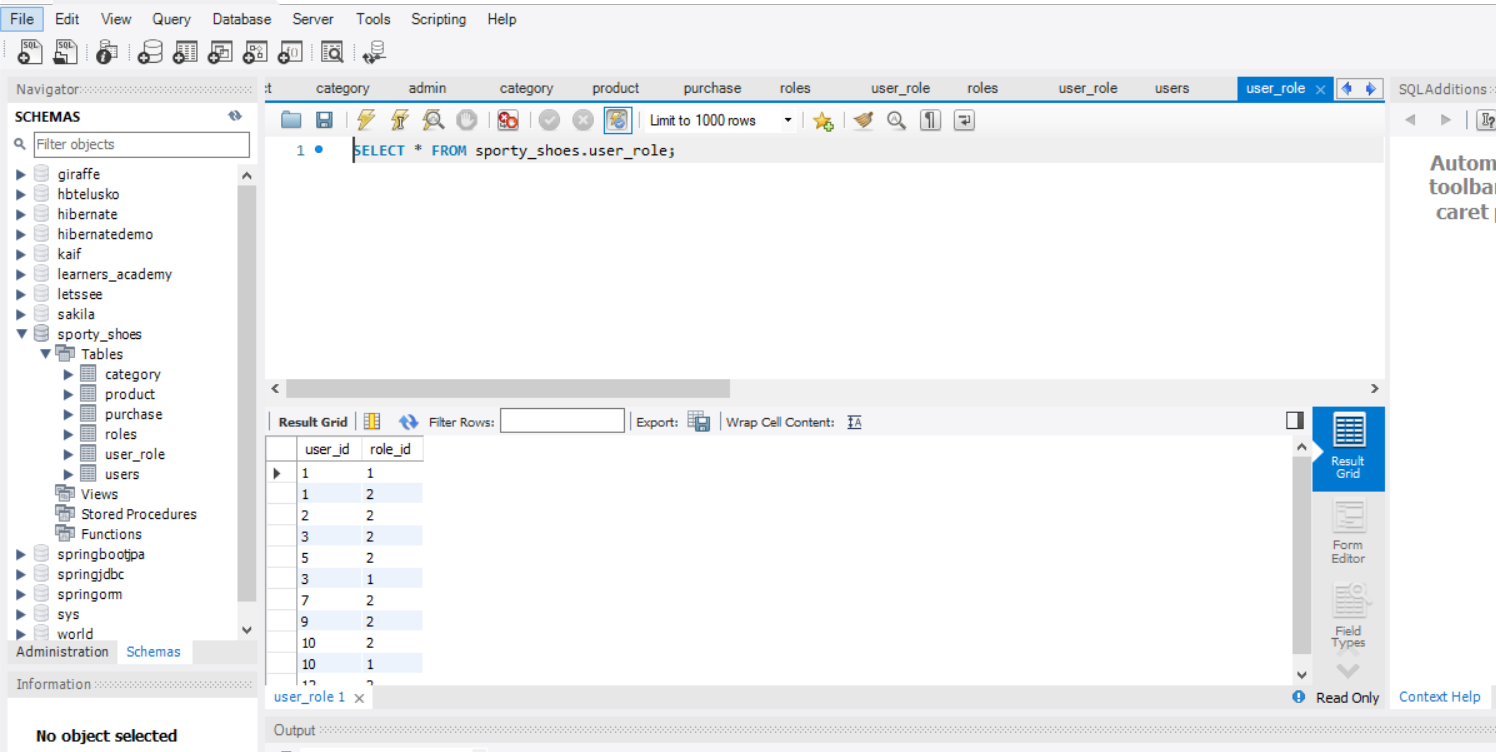
Role entity



User Entity

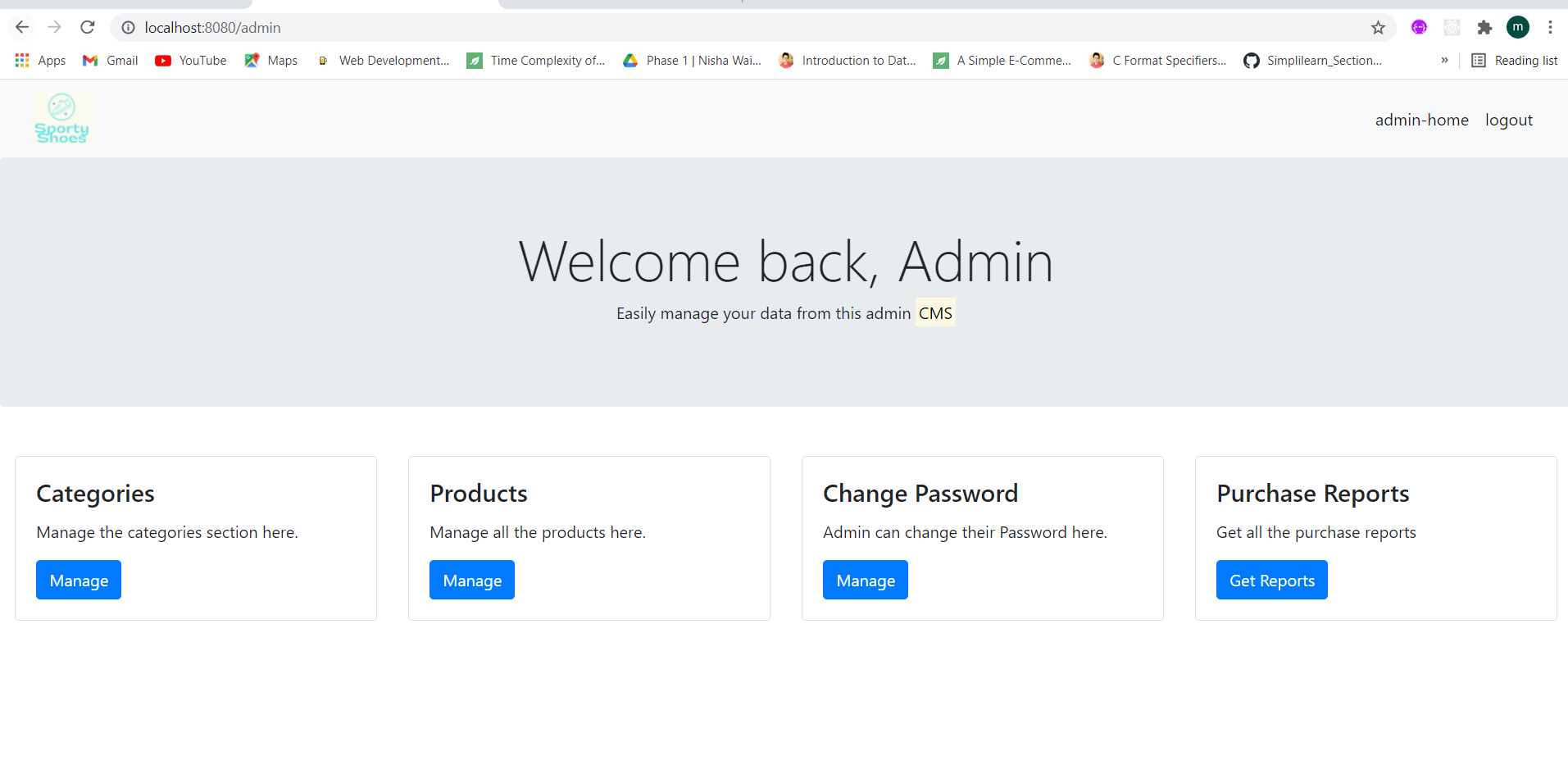


User\_Roles Entity

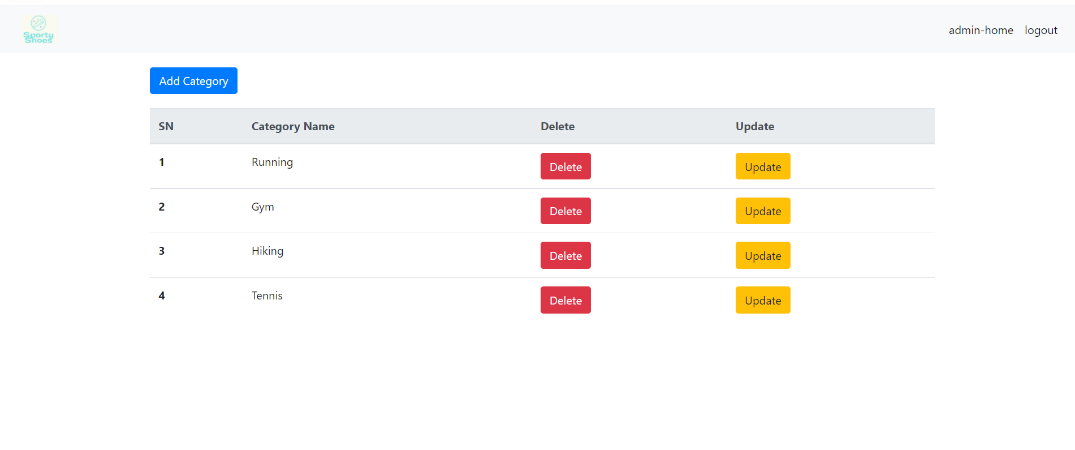


**Screenshots of UI**

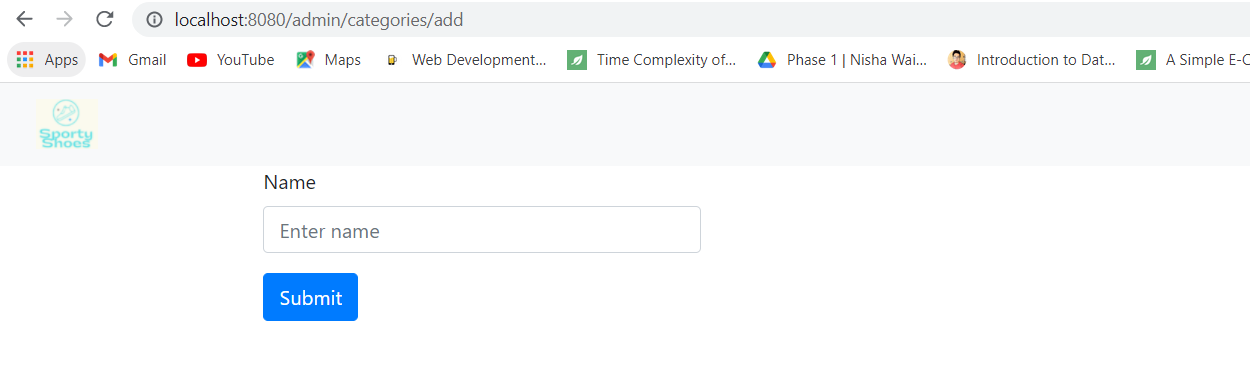
Admin cms

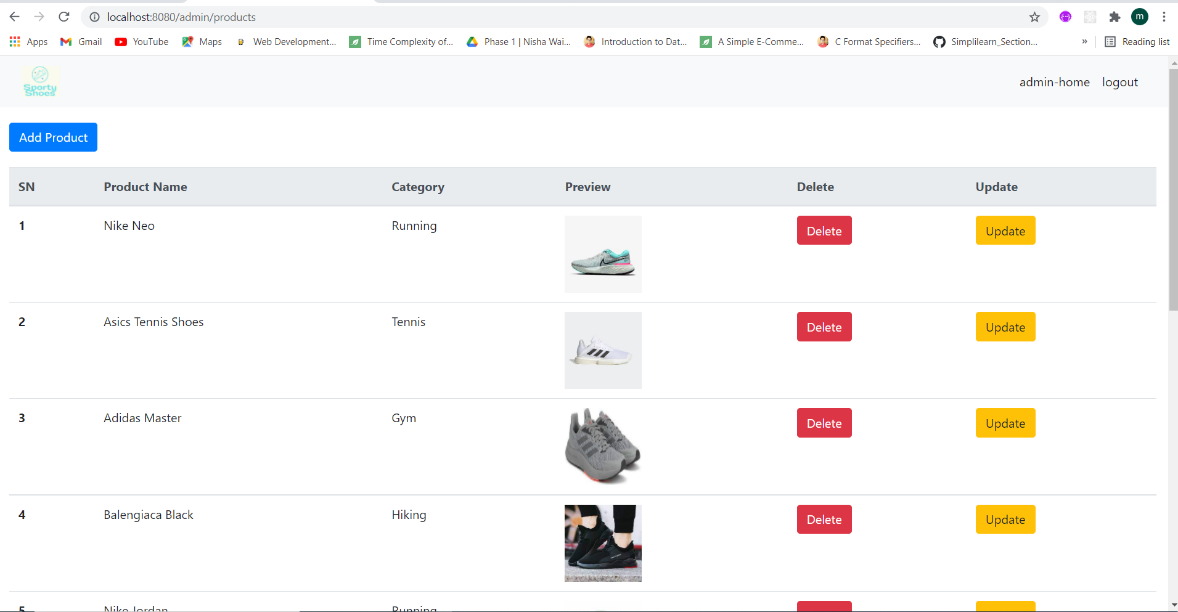
****

Category management

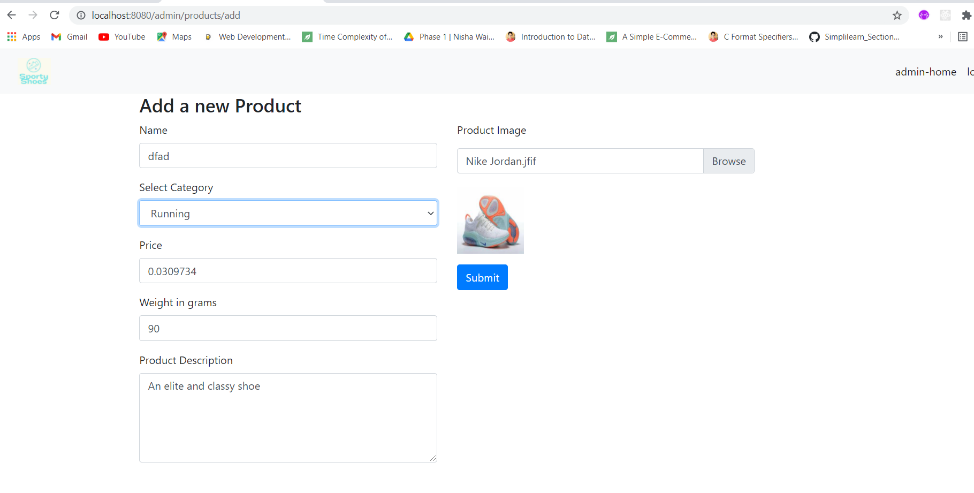


Category add

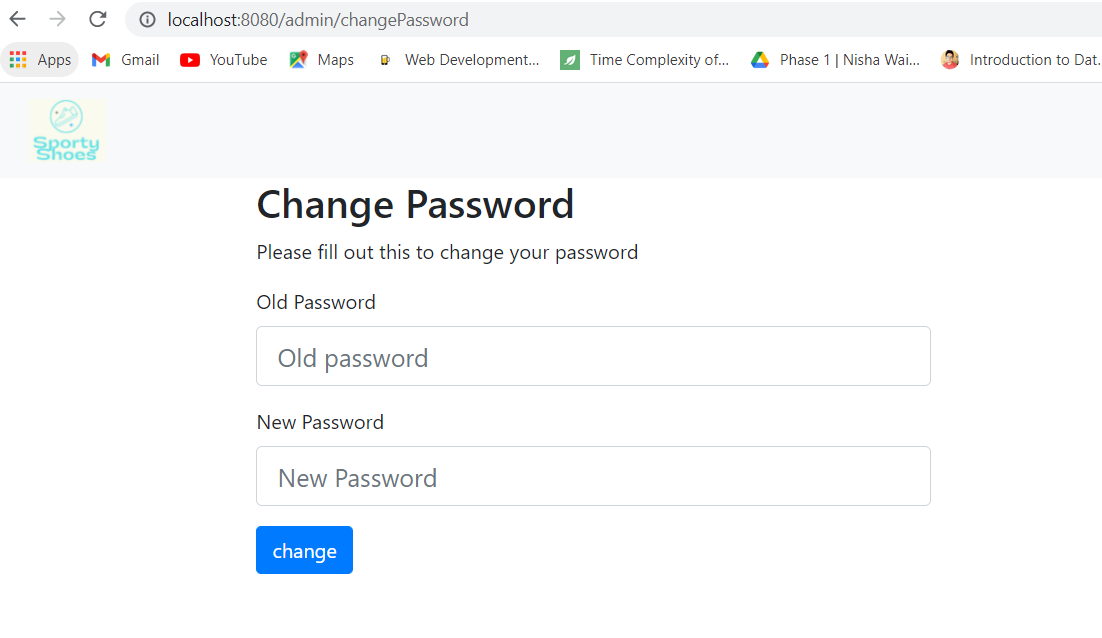
****

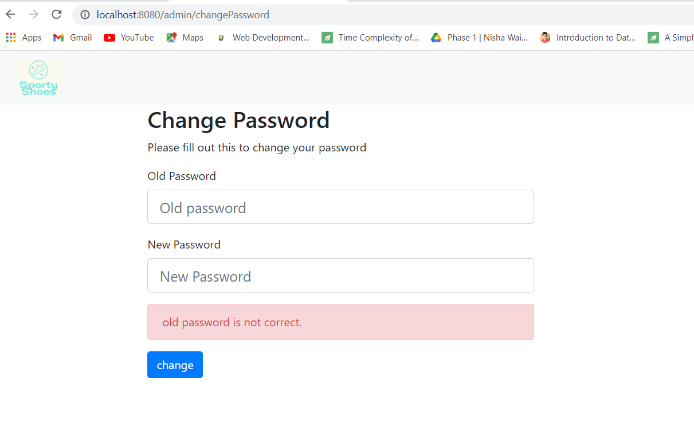
Product Management

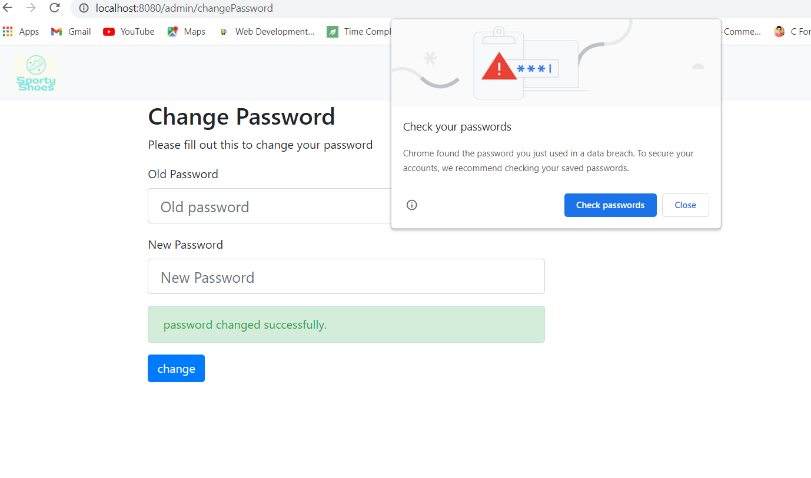
Product add



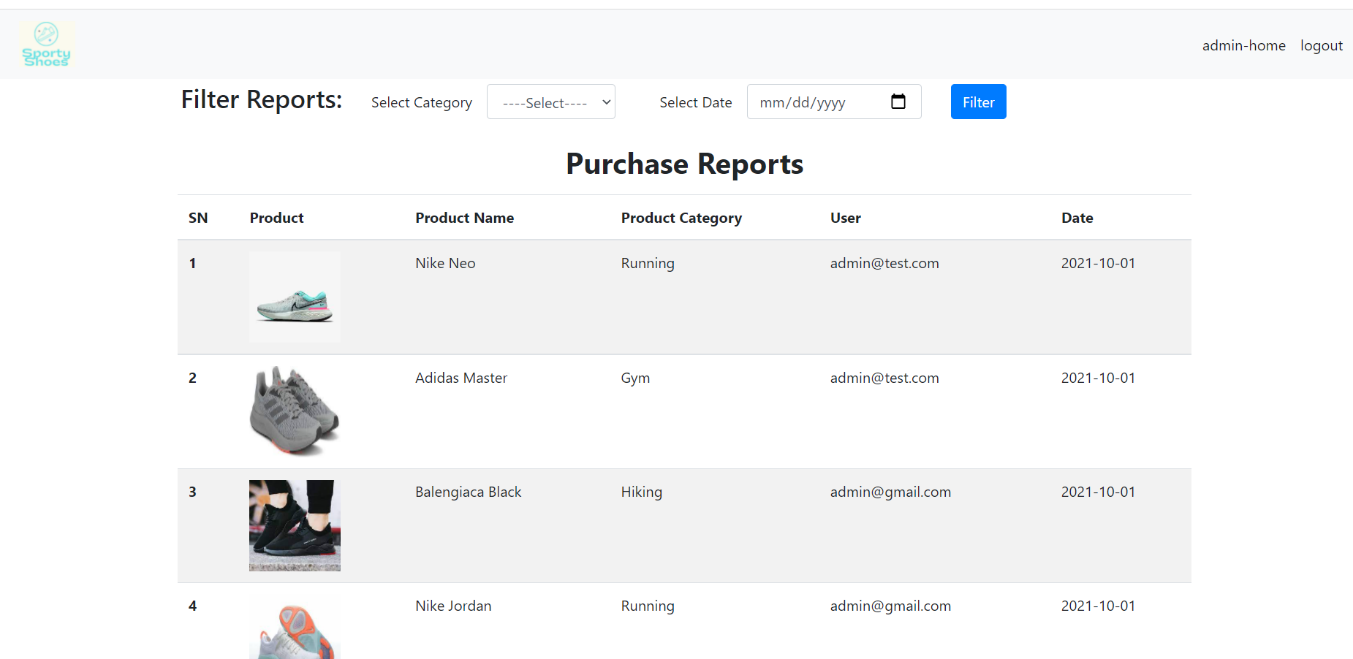
Admin change password

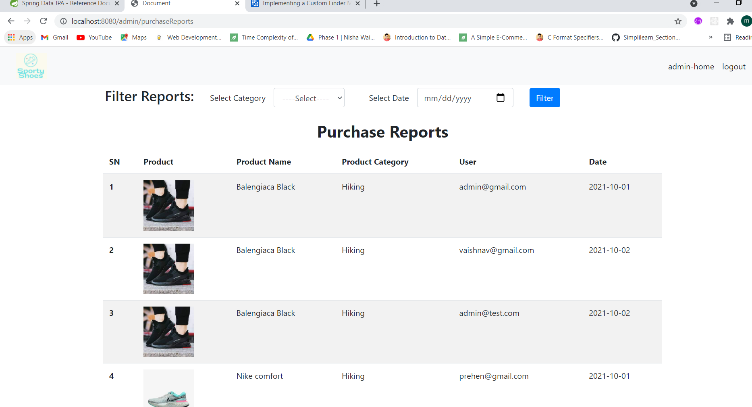
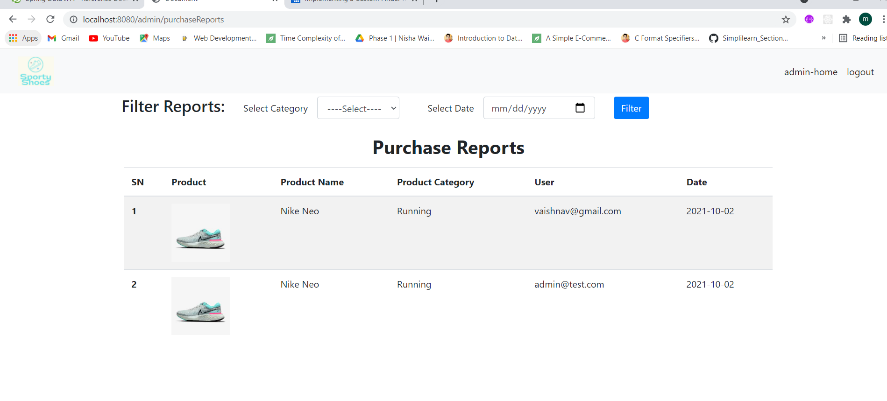


Admin change password validations

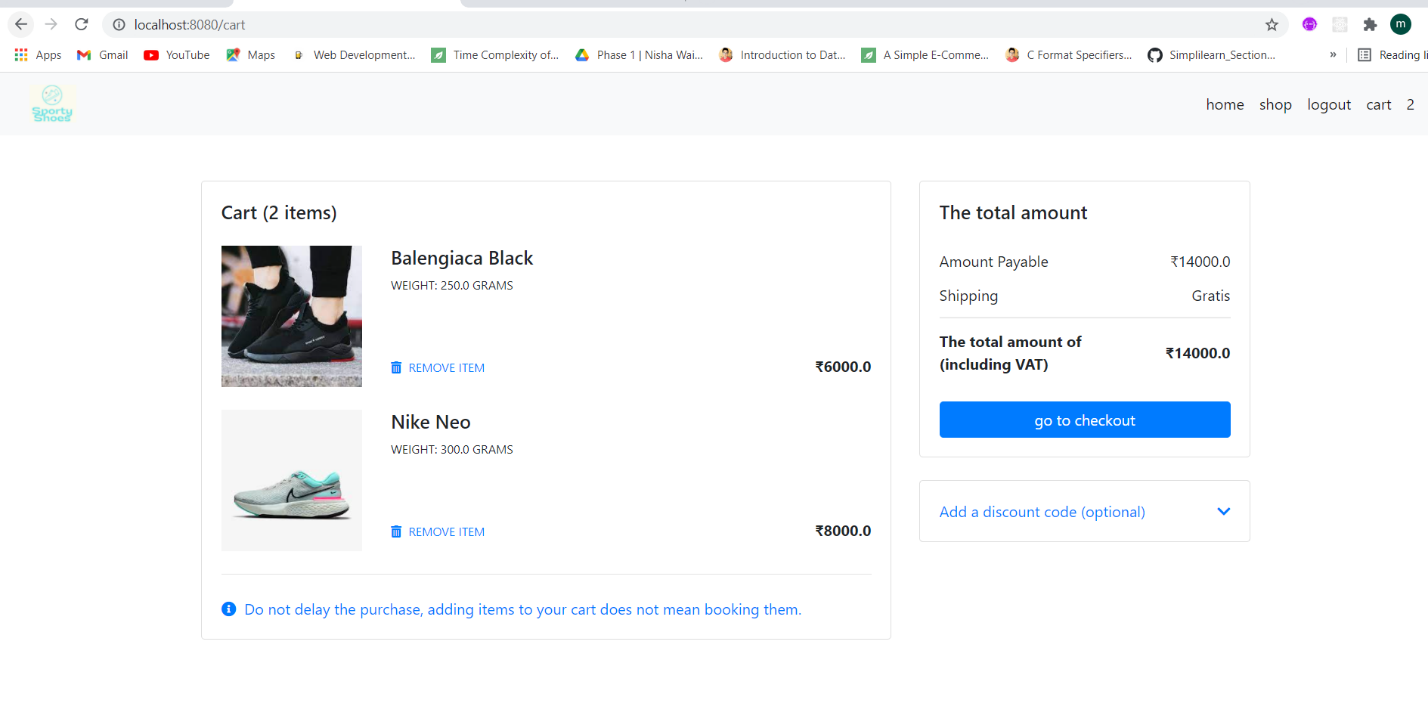


Admin Purchase reports

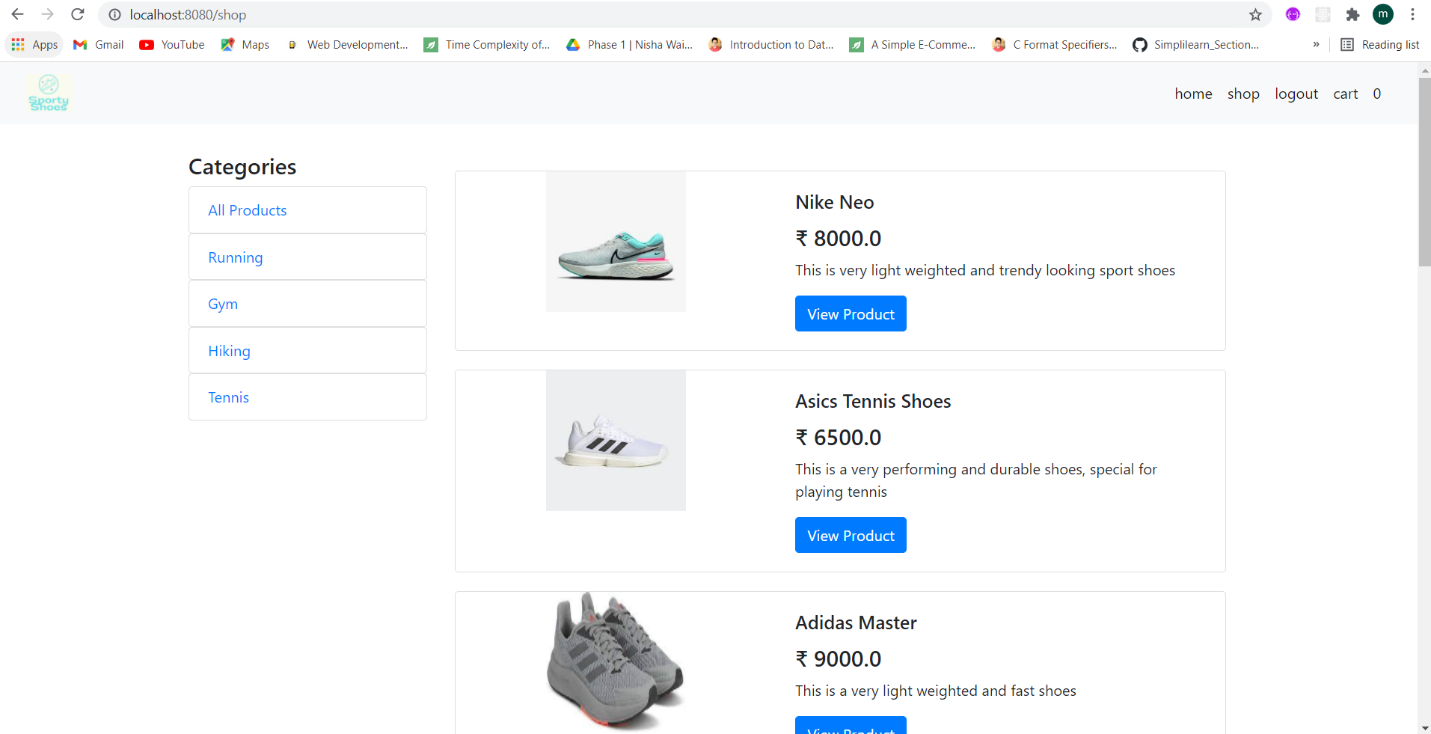


Admin filtered purchase reports

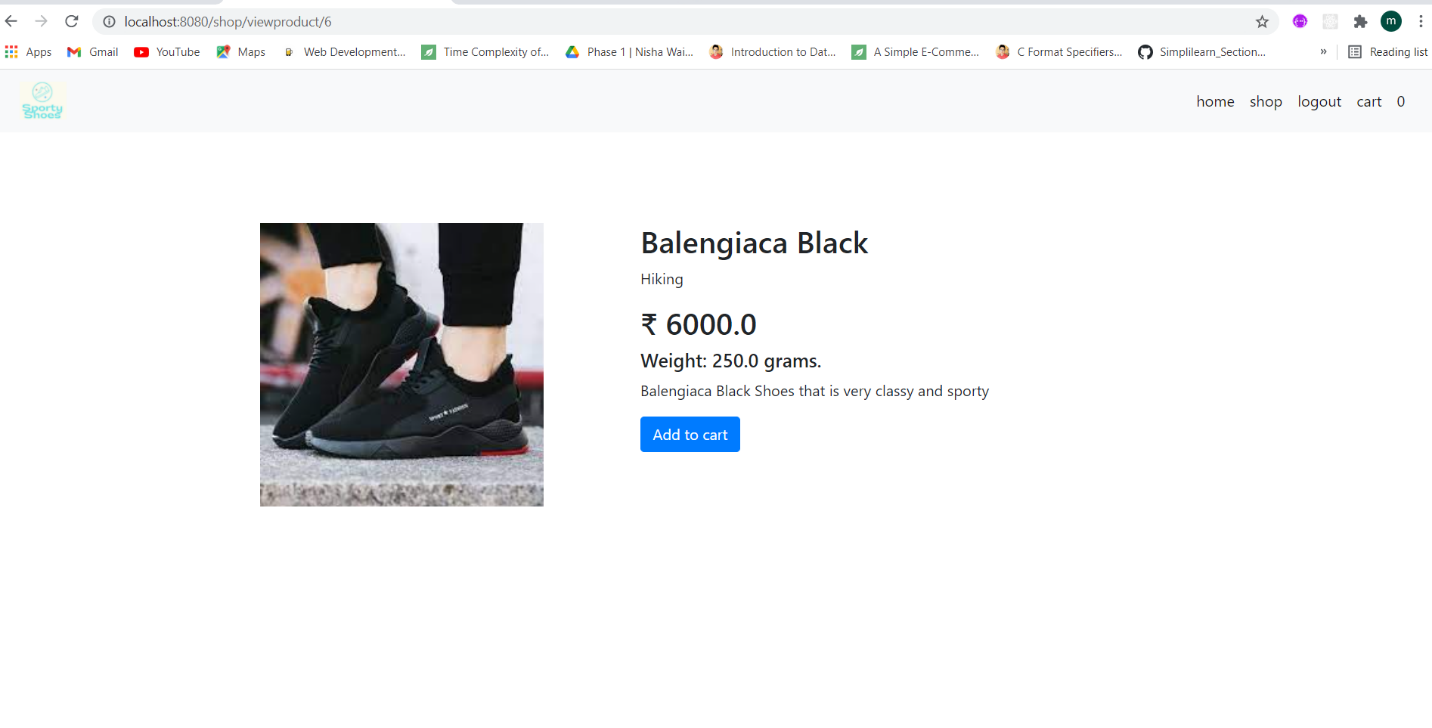
Cart

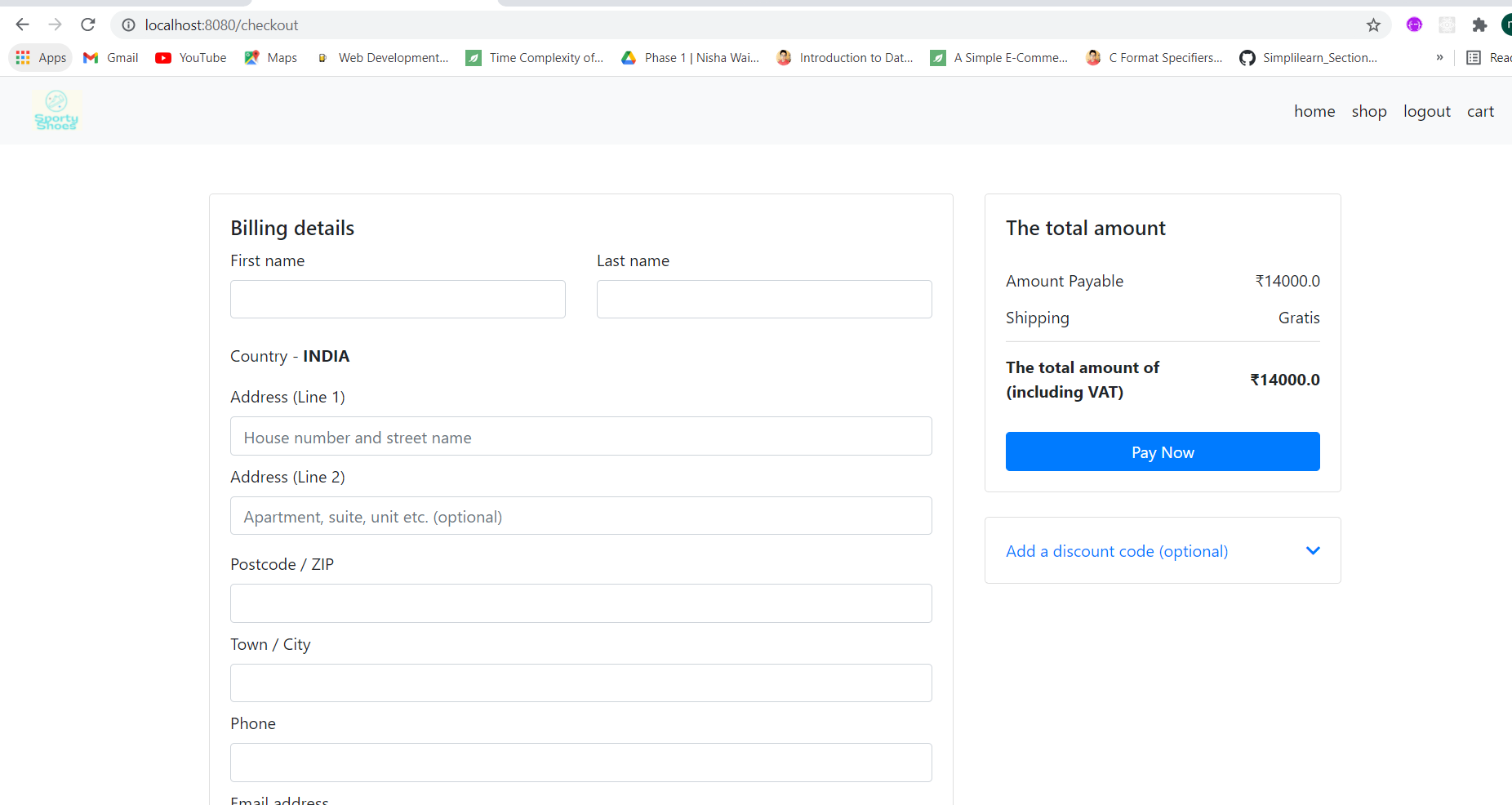


Shop

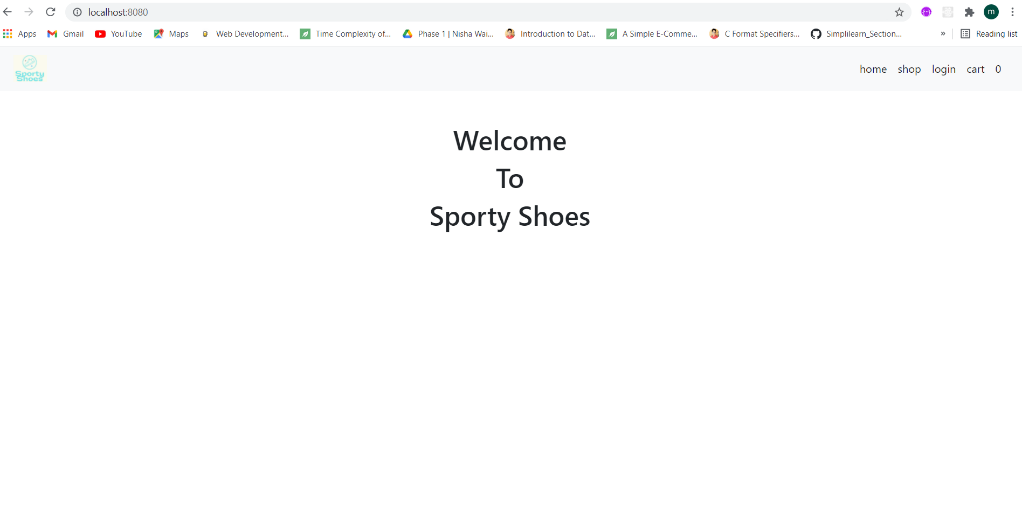


View Product

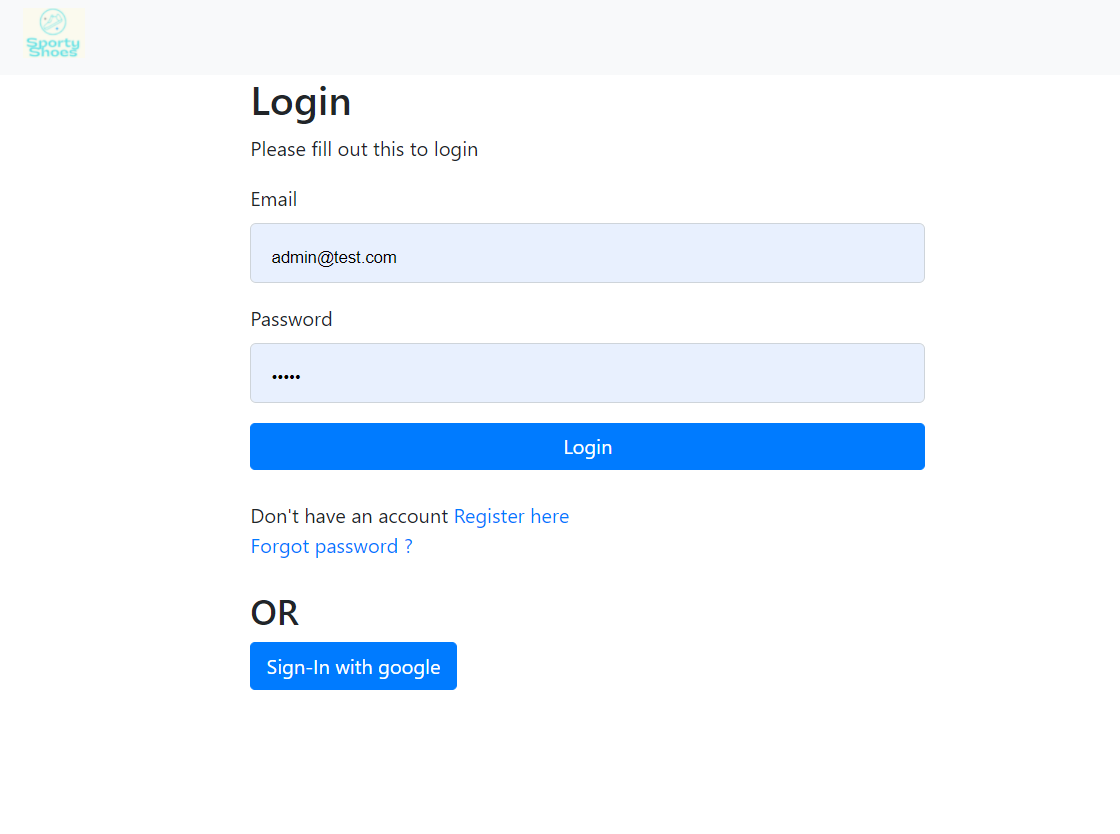


Checkout Page

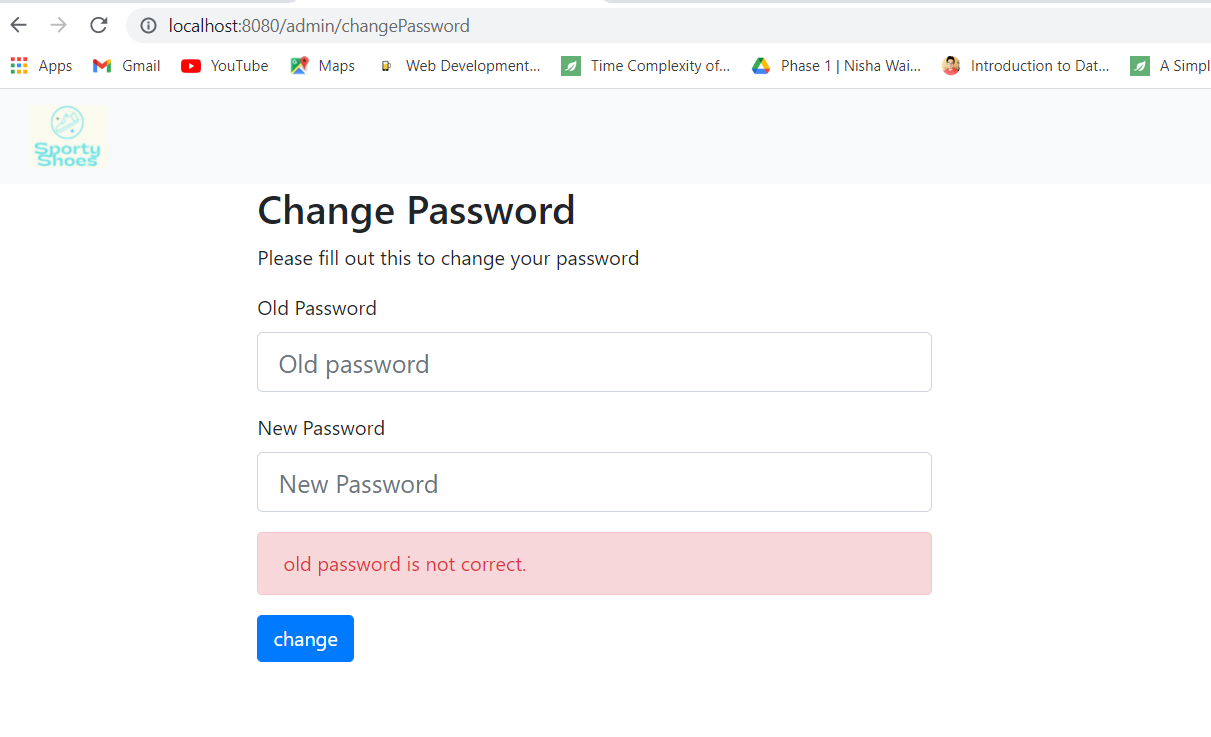
Home Page



Login Page



Login Validation



Register Page

