Development of a Full Stack ecommerce Web Application using Spring Boot, React, Hibernate and MySQL

A Project

Presented to the
Faculty of
California State University Dominguez Hills

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science

in

Computer Science

---

by

GNEY PATEL

SPRING 2023

211860131

Acknowledgement

I would like to thank California State University Dominguez Hills and all the supportive staff and faculty for offering me the chance to accomplish my master's degree in computer science in which I was able to improve both academically and personally. I would like to express my great appreciation to **Dr. Brad Hollister**, my graduation project committee chair, for his advice, patient guidance, enthusiastic encouragement and useful critiques of my graduation project. I would also like to thank **to Dr. Sahar Hooshmand** and **Dr. Bin Tang**, my committee members, who have been instrumental in helping me achieve academic success.

GNEY PATEL

211860131

TABLE OF CONTENT

LIST OF FIGURES

Abstract

The backend of the web application, created with Spring Boot and Hibernate, will offer RESTful APIs to the React-built frontend. A MySQL database will be used by the application to store and retrieve data. Users will be able to communicate with the backend using the frontend's user-friendly interface and RESTful APIs. React, a well-known JavaScript toolkit for creating user interfaces, will be used to develop the user interface. React will make it possible to build reusable components that are simple to include into online applications. Spring Boot, a strong and lightweight framework for creating web applications, will be used to develop the backend. The object-relational mapping (ORM) framework for managing the data persistence layer will be Hibernate. A variety of APIs will be made available by the backend so that the frontend may retrieve and work with data. The application's data will be kept in the MySQL database. The persistent layer will be managed using Hibernate, which makes it simple to bind Java objects to database tables. The database structure will be created to accommodate the requirements of the application, guaranteeing effective data storage and speedy retrieval. This project will show how to create a complete stack web application utilizing cutting-edge tools and industry standards. A fully working web application that enables users to communicate with a MySQL database through an intuitive interface will be produced at the project's conclusion.

CHAPTER 1 INTRODUCTION

The development of a full stack web application using Spring Boot, React, Hibernate, and MySQL is a challenging but rewarding task. In this report, we will discuss the development process, challenges, and solutions involved in creating a robust and scalable web application using these technologies.

Spring Boot is a popular Java-based framework for building web applications. React is a popular front-end library that allows developers to build responsive and dynamic user interfaces. Hibernate is a powerful object-relational mapping framework that enables developers to map Java objects to relational database tables. MySQL is an open-source relational database management system that provides a reliable and scalable solution for storing and managing data.

The combination of these technologies provides a powerful and efficient platform for developing full stack web applications. Spring Boot provides a flexible and extensible architecture for building RESTful web services, while React provides a rich set of components and tools for building dynamic user interfaces. Hibernate simplifies the data access layer by providing an easy-to-use ORM framework, and MySQL provides a scalable and reliable solution for storing and managing data.

This report will cover the design and implementation of a full stack web application using these technologies. We will discuss the architecture of the application, the tools and technologies used, the challenges faced during development, and the solutions implemented to overcome those challenges. Additionally, we will provide a detailed analysis of the performance and scalability of the application, and the lessons learned during the development process.

CHAPTER 2 OVERVIEW

2.1 what is Full Stack Web Application

A full-stack web application refers to a software system that includes both a front-end user interface and a back-end server-side component. It is designed to run on the web, which means it can be accessed via a web browser.

The front-end part of a full-stack web application is responsible for presenting the user interface and handling user interactions. It usually consists of HTML, CSS, and JavaScript code that runs in the user's web browser. The front-end interacts with the back-end via HTTP requests and responses, which enable it to send data to the server and receive data from it.

The back-end part of a full-stack web application is responsible for processing requests, retrieving data from databases, and performing other server-side operations. It is typically built using programming languages such as Java, Python, Ruby, or JavaScript. The back-end interacts with the front-end via APIs (Application Programming Interfaces), which allow it to send and receive data.

Together, the front-end and back-end components of a full-stack web application enable users to interact with the application through a web browser, and enable the application to store and retrieve data from a database or other sources. This makes it possible to build complex, interactive web applications that can be accessed from anywhere with an internet connection.

2.2 what is Spring Boot

Spring Boot is a popular Java-based framework used for developing standalone, production-grade web applications quickly and easily. It is built on top of the Spring Framework and provides a streamlined way to develop web applications with minimal setup and configuration.

Spring Boot includes many features that simplify application development, such as auto-configuration, which automatically configures the application based on the classpath, and embedded servers, which allow the application to be run without the need for a separate web server. It also includes a powerful command-line tool that can be used to create, run, and manage Spring Boot applications.

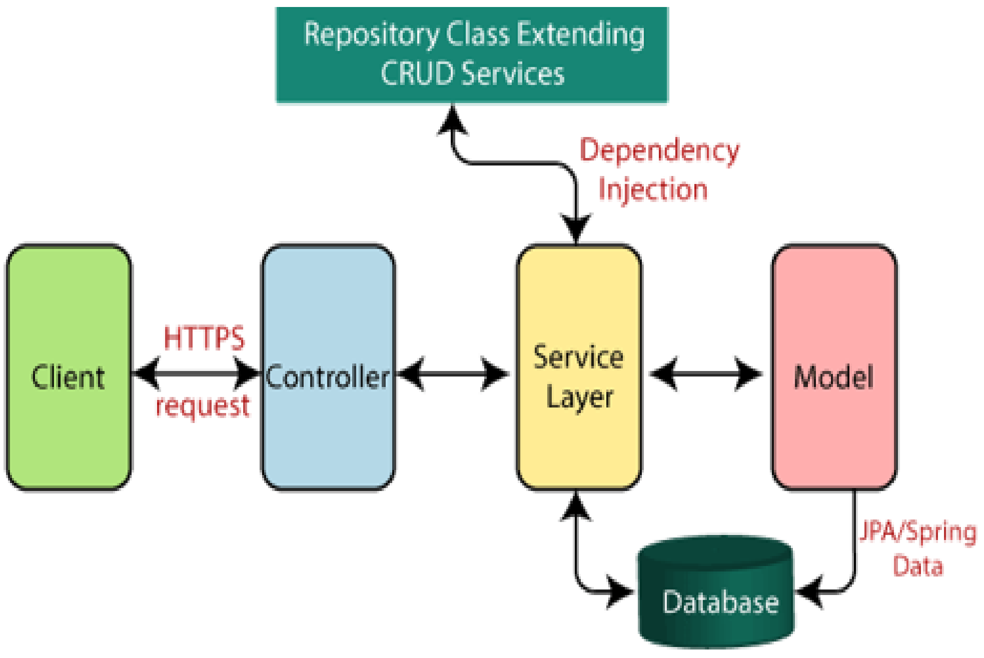


Figure.1 Spring architecture[1]

A DSL (Domain-Specific Language) called CRUDyLeaf expands the DSL programming languages used to create and design in a way that is based on the issue that arose in addressing the domain with a variety of potential solutions [2]. Additionally, the spring boot framework creates CRUDyLeaf database records for CRUD entity operations using an external API called "REST"

Spring Boot supports a wide range of technologies and frameworks, including web applications, databases, messaging systems, and more. It also has a large and active community of developers, who contribute to the project and provide support and resources to other developers.

Overall, Spring Boot is a popular choice for developers who want to quickly and easily develop web applications using Java.

2.3 what is React

React was introduced to the world two years ago, and since then it has seen impressive growth, both inside and outside Facebook. [3] New web projects at Facebook are commonly built using React in one form or another, and it is being broadly adopted across the industry. [3] Developers and engineers are choosing React because it allows spending more time to focus more on the product development and less time spent on fighting and learning to the framework. A React application is a collection of discrete components, each representing a single view.

The idea of every individual view component makes it easy to iterate on product development because to make changes on a single view or component, it is not necessary to consider the entire system. When an application is built with React, the code is generally predictable, it is because React wraps the DOM mutative, imperative API with a declarative one, which raises the level of abstraction and simplifies the programming model. [3] Moreover, it is easier to scale the application built with React. The combination of React and the rapid iteration cycle of the web, has enabled to make some excellent products including many Facebook components. An amazing JavaScript framework called Relay has also been made on top of React, which helps simplifying data fetching on a large scale. [3]

React also uses a virtual DOM (Document Object Model), which is an in-memory representation of the actual DOM used by browsers. By manipulating the virtual DOM, React can efficiently update the actual DOM, reducing the number of updates needed and improving performance.

React can be used to build a wide range of applications, from small single-page apps to large-scale enterprise applications. It is often used in conjunction with other technologies and frameworks, such as Redux for state management, React Native for building mobile apps, and Next.js for server-side rendering.

2.4 what is API

API stands for Application Programming Interface. It is a set of protocols, routines, and tools for building software applications. An API defines how software components should interact with each other, simplifying the process of software development and enabling different applications to communicate with each other.

In recent years, API has been widely recognized in businesses and web services. All industries use APIs or application program interfaces as a vital tool in securely sharing the data and information to other services. It allows enterprises to innovate and develop new ideas by sharing existing data and information. Several API types emerged, one of which is REST or Representational State Transfer [4] [5]. REST or REST API is a web service architecture that offers a communication link between online platforms or systems. REST API is a norm used to build a network-based software system as an architectural instrument. In REST API, data are being transferred using the HTTP Protocol that is costeffective in terms of internet traffic and in a way that is easy to use.

APIs can be used to access data from third-party applications, such as social media platforms or financial data providers, to integrate different systems or to create custom software solutions. They often allow developers to access functionalities of an application or a service by making requests to its endpoints.

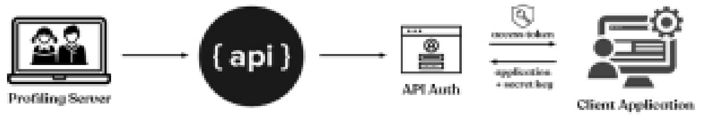


Figure 2.API process

[6]The graphical depiction of the API request procedure is shown in Figure 2. The administrator will give the client application an encrypted application and a secret key.

[6]The client application will use the application and secret key as its specific login information while making API requests in order to use API Auth. The request will be processed to the API by retrieving student data from the profiling server after the provided credentials were verified by the API Auth. The API Auth will deliver the requested information together with the access token after the necessary information was ready to be issued.

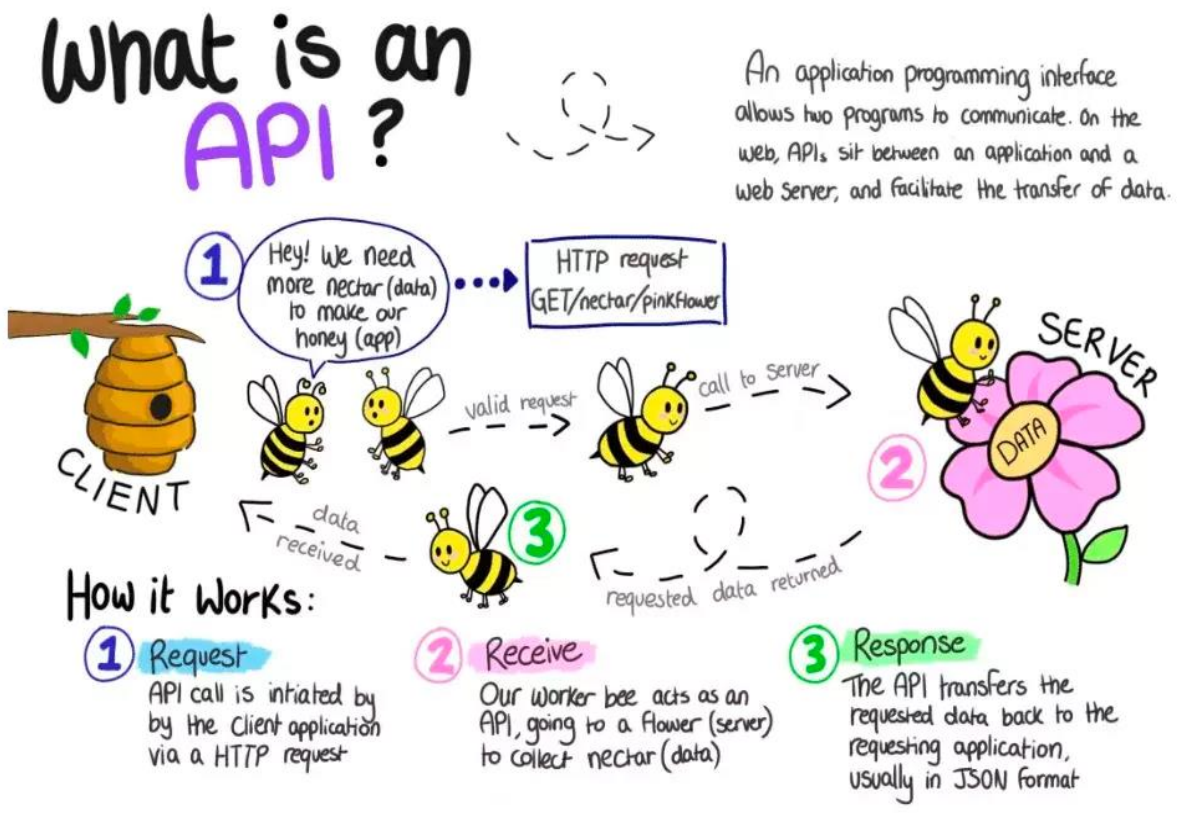

What is an API ?
An application programming interface allows two programs to communicate. On the web, APIs sit between an application and a web server, and facilitate the transfer of data.
1
Hey! We need more nectar (data) to make our honey (app)
HTTP request GET/nectar/pinkflower
CLIENT
valid request
call to server
SERVER
DATA
2
3
data received
requested data returned
How it Works:
1 Request
API call is intiated by by the Client application via a HTTP request
2 Receive
Our worker bee acts as an API, going to a flower (server) to collect nectar (data)
3 Response
The API transfers the requested data back to the requesting application, usually in JSON format

Figure 3.API picture formation

APIs can be built using different programming languages and architectures, but they all have the common goal of providing a standardized way for software components to interact with each other. They can be private, meant for internal use within an organization, or public, meant to be used by external developers to create new applications or services.

Overall, APIs play a crucial role in modern software development by enabling seamless communication and integration between different applications, services, and platforms.

2.5 what is Hibernate MySQL

Hibernate is a project focused on managing persistent data in Java. The Hibernate suite includes projects as Hibernate ORM, Hibernate EntityManager, Hibernate Search, Hibernate Validator, Hibernate OGM, Hibernate Envers, or Hibernate Reactive [7].

Hibernate ORM belongs to the larger Hibernate framework suite. It provides a native proprietary API and a core for persisting information in databases. Hibernate ORM is at the ground of other projects belonging to the same suite. It is an independent framework and may be used on any modern Java version and eventually in combination with other frameworks (including here Spring and Jakarta Enterprise Edition) [7].

Rather than utilizing bytecode processing or code generation, hibernate uses runtime reflection to determine the persistent properties of a class. The objects to be persisted are defined in a mapping document, which serves to describe the persistent fields and associations, as well as any subclasses or proxies of the persistent object. The mapping documents are compiled at application startup time and supply the framework with necessary information for a class [8]. In addition to it, they are used in support operations, such as generating the database schema or creating stub and Java source files.

A Session Factory is created from the compiled collection of mapping documents[7]. The Session Factory provides the mechanism for managing persistent classes and the Session interface. The Session class provides the interface between the persistent data store and the application. The Session interface wraps a JDBC connection, which can be usermanaged or controlled by hibernate, and is only intended to be used by a single application thread, then closed and discarded.

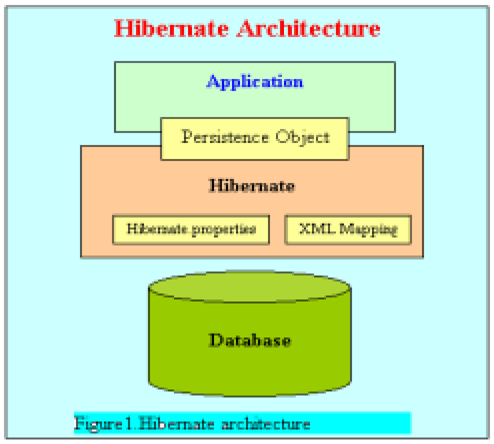


Figure 4.Hibernate Architecture

The following Figure4 describes the high level architecture of hibernate i.e. they show how hibernate uses the database and configuration data to provide persistence services (and persistent objects) to an application. To use Hibernate, it requires creating Java classes that represent the table in the database and then map the instance variable in the class with the columns in the database. Then, Hibernate can be used to perform operations on the database like select, insert, update and delete the records in the table. Hibernate automatically creates the query to perform these operations. The Figure4 describes the high level architecture of hibernate.

Hibernate architecture has following three main components. Connection management: Hibernate Connection management service grant efficient management of the database connections. Database connection is the priciest part of database as it requires a lot of resources of open/close the database connection. Transaction management: Transaction management service provides the capability to the user to execute more than one database statements at a time. Object relational mapping: Object relational mapping is a technique of mapping the data representation from an object model to a relational data model. This part of the hibernate is used to select, insert, update, view and delete the records form the underlying table. When we pass an object to a Session.save() method, hibernate reads the state of the variables of that object and executes the necessary query. Hibernate is extremely good tool as far as object relational mapping is concern, but in terms of connection management and transaction management, it lacks in performance and capabilities. So usually hibernate is being used with other connection management and transaction management tools. For example apache DBCP is used for connection pooling with the hibernate. Hibernate provides a lot of flexibility in usage. It is called "Lite" architecture when we only use object relational mapping component. While in "Full Cream" architecture all the three component Object Relational mapping, Connection Management and Transaction Management are used.

CHAPTER 3 METHODOLGOY AND IMPLEMENTATION

3.1Architecture

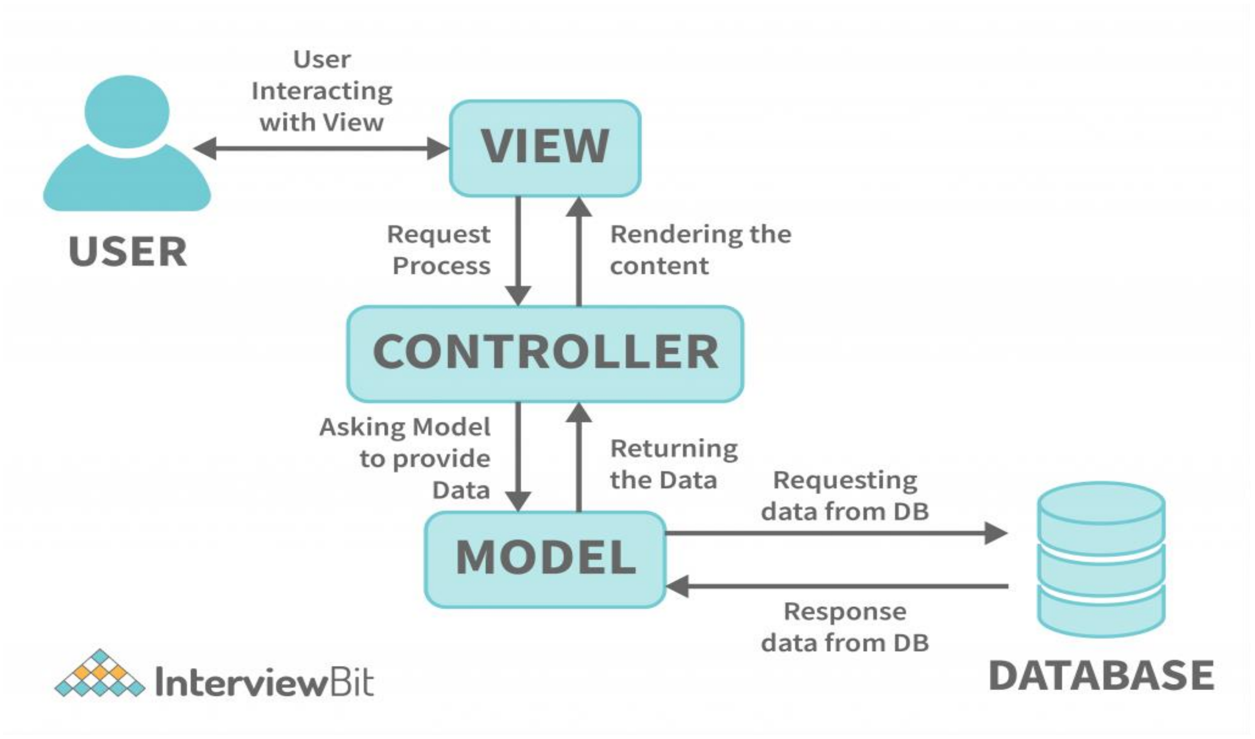


Figure 5.MVC architecure

Model view controller (MVC) [9] is an architectural pattern usually used in web-based applications. It provides three main layers; model, view, and controller. Many developers use MVC as a standard design pattern. It is a complete framework. MVC provide three types of classes:

A. Model- Model classes are used to implement the logic of data domains. These classes are used to retrieve, insert or update the data into the database associated with our application.

B. View- Views are used to prepare the interface of our application. By using that interface users interact with our application.

C. Controller- Controller classes are used to respond to the user's requests. Controller classes perform the users requested actions.

These classes work with model classes and select the appropriate view that should be displayed to the user according to user requests. MVC pattern architecture is basically a three-layered architecture. It separates the characteristics of application. Its first layer is related to the user input logic, second layer is related to the business logic and third layer is used to implement user interface logic. MVC provide very loose coupling among these three layers. MVC pattern are used to specify the location of each logic in application [10].

MVC patterns provide the facility of parallel development. It means that each layer of the application independent of each other i.e. three developer can work on the single application simultaneously [11]. One developer will be working on user input logic (controller logic), other developer will be working on the user interface logic (view) and third developer will be working on the business logic (model) at the same time.

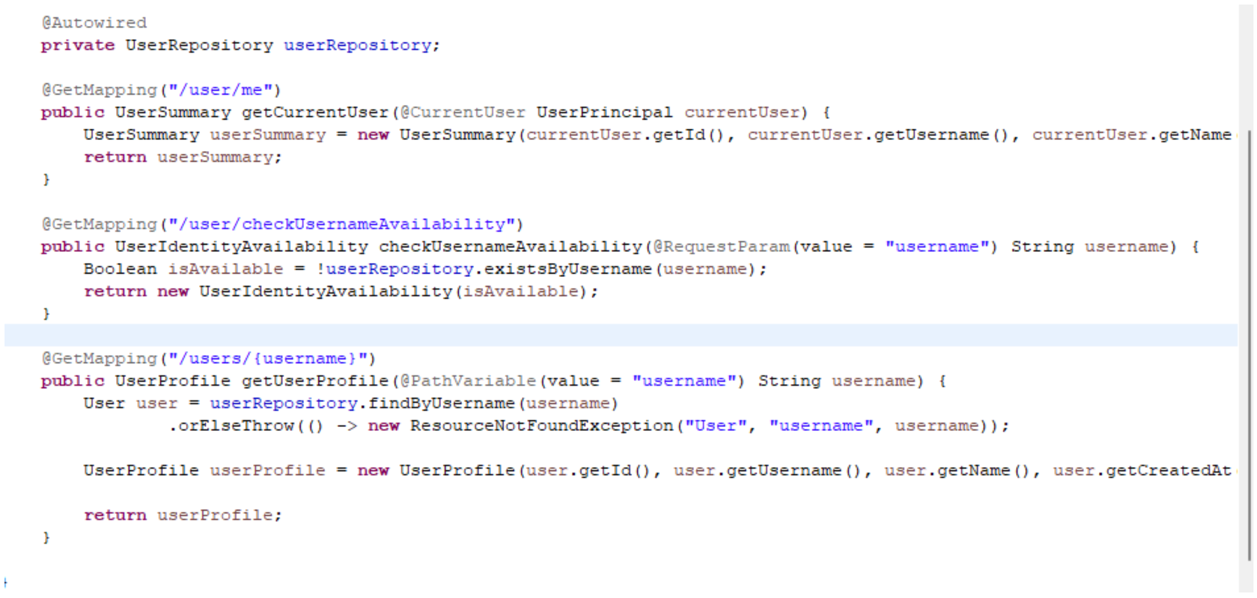
3.2 Overview of software

- User registration and authentication: Allow users to create accounts, log in securely, and manage their profiles.
- Product catalog: Display a wide range of products with relevant details, including images, descriptions, and pricing information.
- Shopping cart: Enable users to add products to their carts, update quantities, and proceed to checkout.
- Order management: Implement features like order history, order tracking, and payment processing.
- Admin panel: Provide an administrative interface for managing products, inventory, and user accounts.
- Search functionality: Implement a search feature that allows users to find products based on keywords or categories.
- Responsive design: Ensure the application works seamlessly across various devices, including desktops, tablets, and mobile phones.

3.3 Result and workflow

The aim is to provide the best functionality to user to get desirable product which user want to purchase with best efficient software.

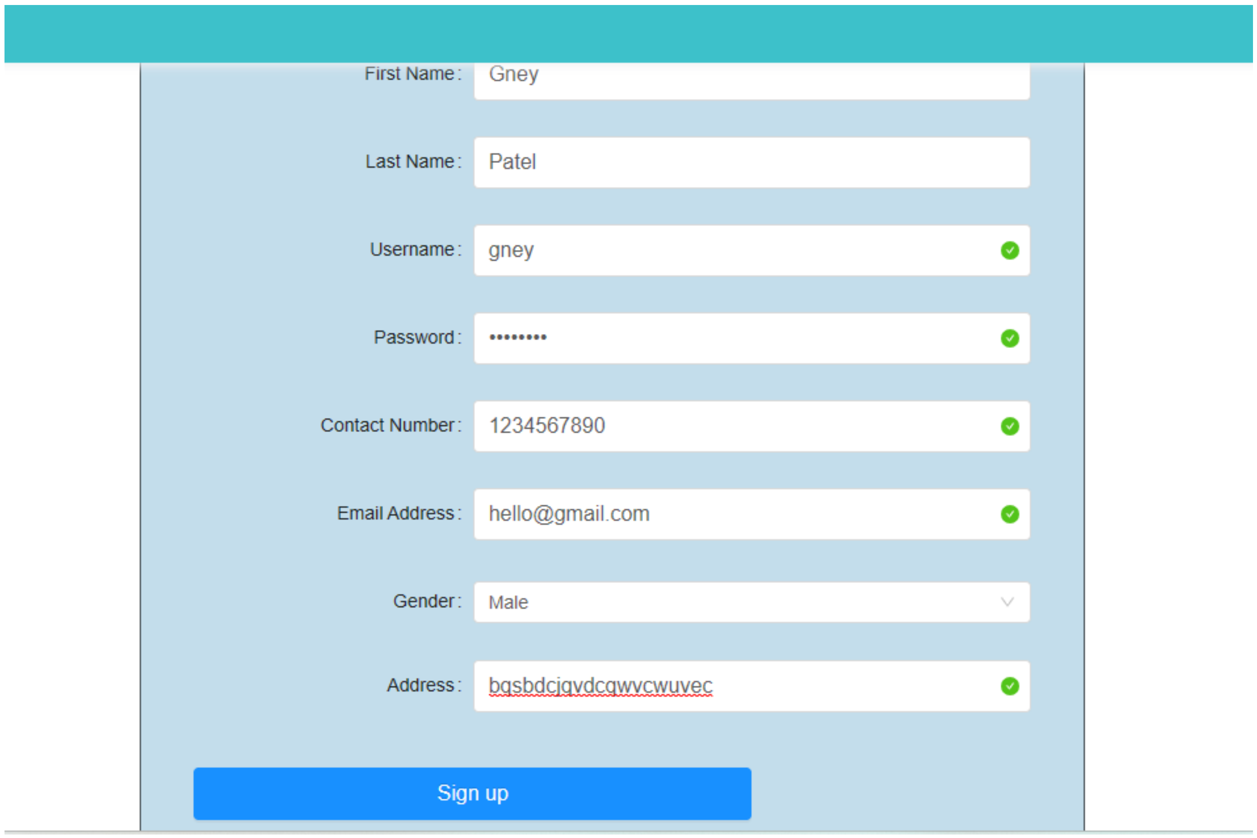
- The user open up the web site and click on register button and user will get message “successfully registered”.(as soon as user hit the register button React application direct to backend part and user controller will handle the event).



```
@Autowired
private UserRepository userRepository;

@GetMapping("/user/me")
public UserSummary getCurrentUser(@CurrentUser UserPrincipal currentUser) {
    UserSummary userSummary = new UserSummary(currentUser.getId(), currentUser.getUsername(), currentUser.getName
    return userSummary;
}

@GetMapping("/user/checkUsernameAvailability")
public UserIdentityAvailability checkUsernameAvailability(@RequestParam(value = "username") String username) {
    Boolean isAvailable = !userRepository.existsByUsername(username);
    return new UserIdentityAvailability(isAvailable);
}

@GetMapping("/users/{username}")
public UserProfile getUserProfile(@PathVariable(value = "username") String username) {
    User user = userRepository.findByUsername(username)
            .orElseThrow(() -> new ResourceNotFoundException("User", "username", username));

    UserProfile userProfile = new UserProfile(user.getId(), user.getUsername(), user.getName(), user.getCreatedAt

    return userProfile;
}
```

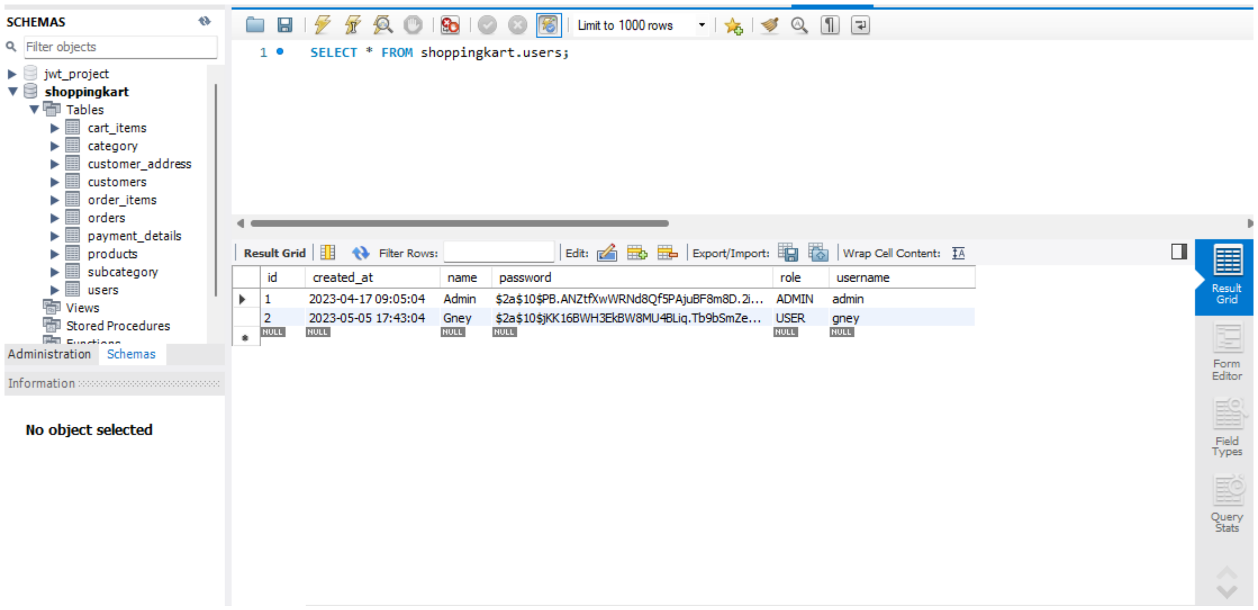
Figure 6.User controller code

- Then after registering his data into database the home screen appears.



First Name:	Gney
Last Name:	Patel
Username:	gney
Password:	••••••••
Contact Number:	1234567890
Email Address:	hello@gmail.com
Gender:	Male
Address:	bqsbdcjqvdcqwvcwuvec

Figure 7.User register page

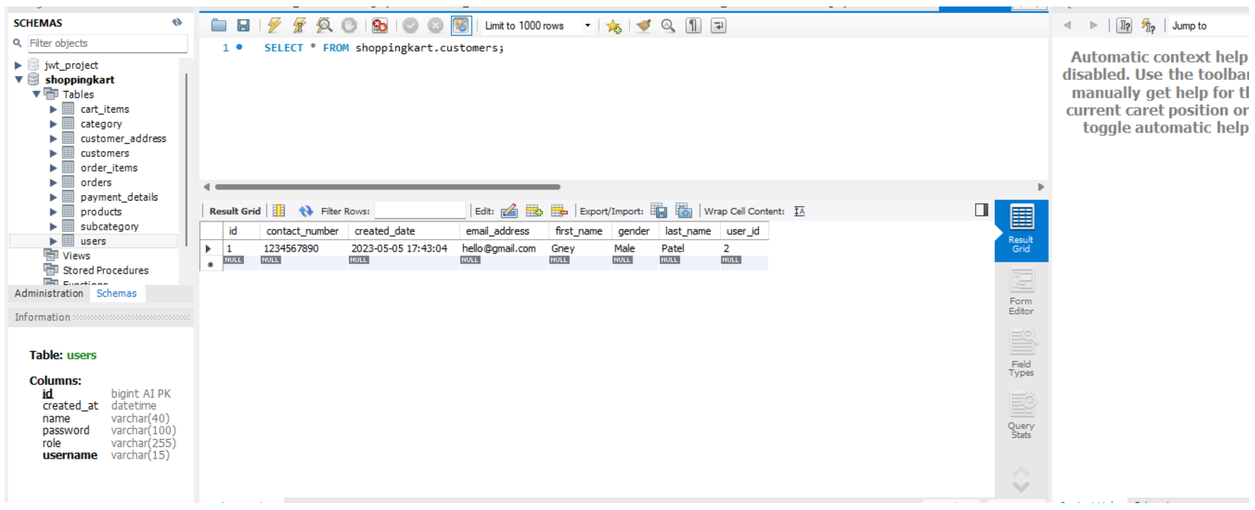


```sql
SELECT * FROM shoppingkart.users;
```

id	created_at	name	password	role	username
1	2023-04-17 09:05:04	Admin	$2a$10$PB.ANZtfXwWRNd8Qf5PAjuBF8m8D.2i...	ADMIN	admin
2	2023-05-05 17:43:04	Gney	$2a$10$jKK16BWH3EkBW8MU4BLiq.Tb9bSmZe...	USER	gney
NULL	NULL	NULL	NULL	NULL	NULL

No object selected

Figure 8. Database for user (describe the role whether it is admin or user)



```sql
SELECT * FROM shoppingkart.customers;
```

Automatic context help disabled. Use the toolbai manually get help for tl current caret position or toggle automatic help

id	contact_number	created_date	email_address	first_name	gender	last_name	user_id
1	1234567890	2023-05-05 17:43:04	hello@gmail.com	Gney	Male	Patel	2
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table: users

Columns:

id	bigint AI PK
created_at	datetime
name	varchar(40)
password	varchar(100)
role	varchar(255)
username	varchar(15)

Figure 9.Database for User details

- Where user can search the product by his category or random product which he wants to purchase.Described in figure 10,11

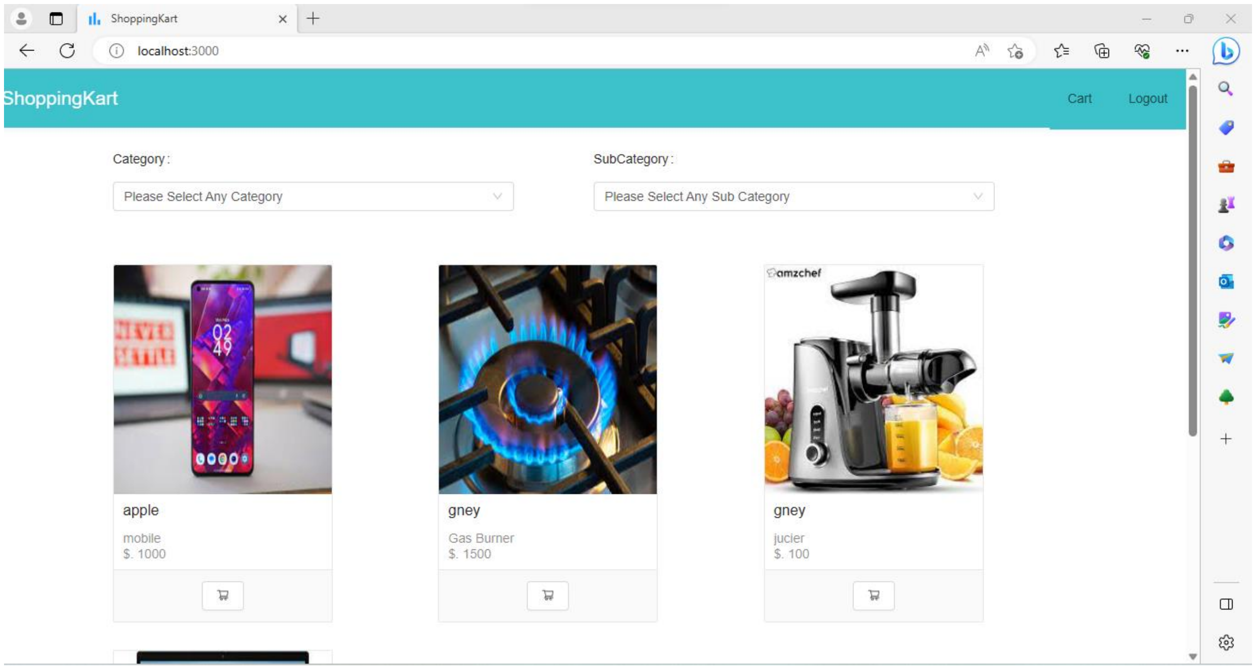


Figure 10.View of home page

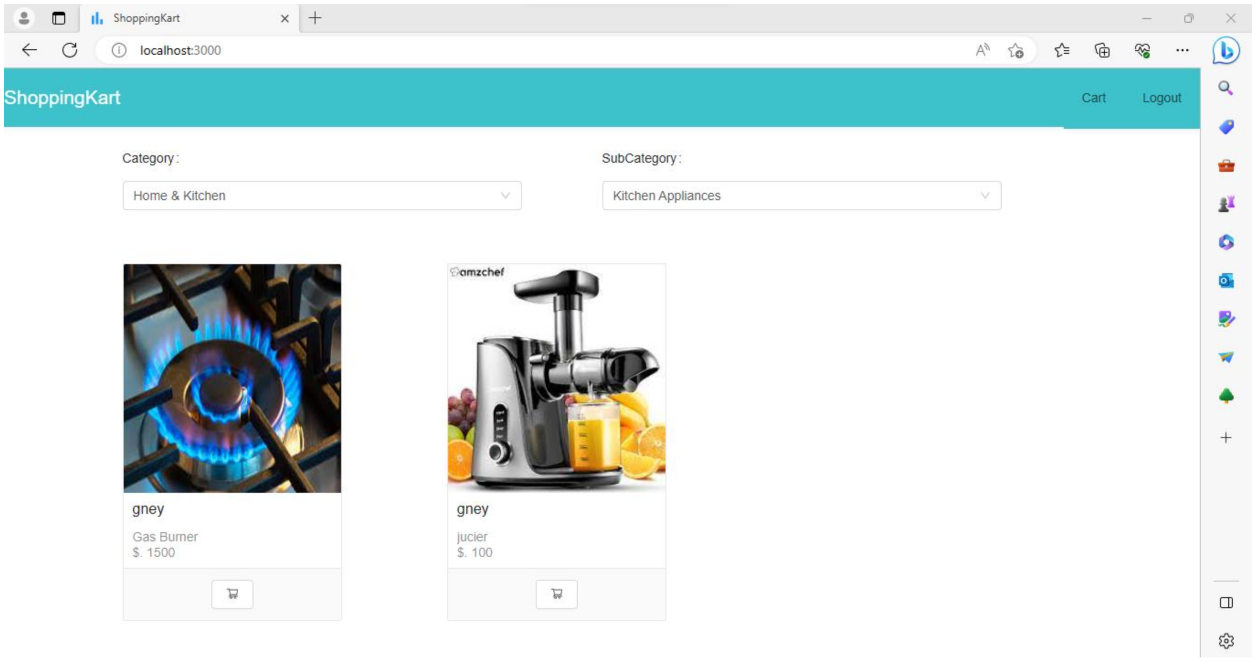
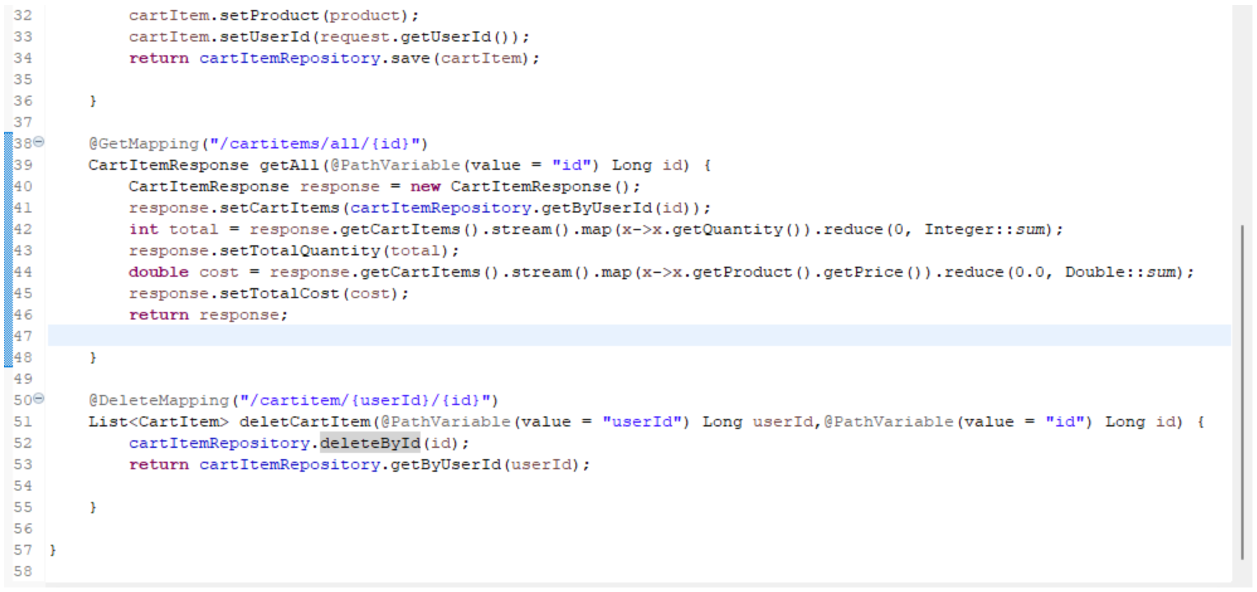


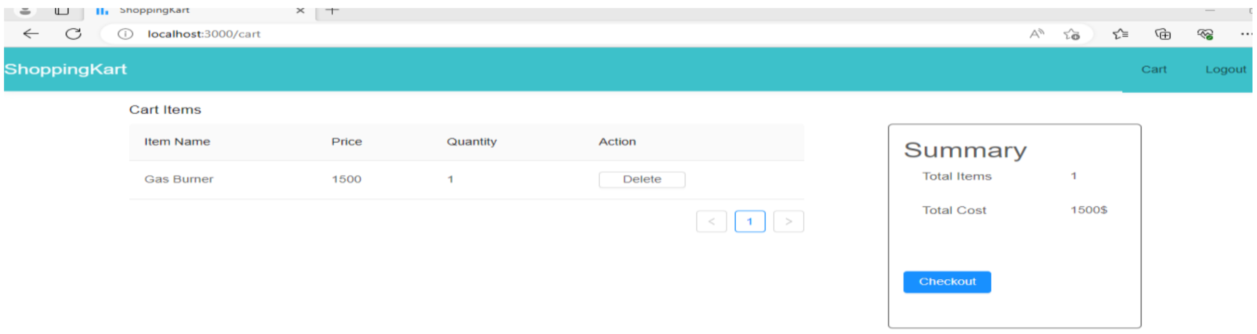
Figure 11 View of selected category home page

- After clicking on the product that particular product will be added into the cart item .(backend:- as soon as cart item button hit the product id will go to cart controller and that id product will fetched from product class and then it will be added to cart new id) Described in figure 13 and figure 12 describe the backend part of cart item



```
        cartItem.setProduct(product);
        cartItem.setUserId(request.getUserId());
        return cartItemRepository.save(cartItem);

    }

    @GetMapping("/cartitems/all/{id}")
    CartItemResponse getAll(@PathVariable(value = "id") Long id) {
        CartItemResponse response = new CartItemResponse();
        response.setCartItems(cartItemRepository.getByUserId(id));
        int total = response.getCartItems().stream().map(x->x.getQuantity()).reduce(0, Integer::sum);
        response.setTotalQuantity(total);
        double cost = response.getCartItems().stream().map(x->x.getProduct().getPrice()).reduce(0.0, Double::sum);
        response.setTotalCost(cost);
        return response;

    }

    @DeleteMapping("/cartitem/{userId}/{id}")
    List<CartItem> deleteCartItem(@PathVariable(value = "userId") Long userId,@PathVariable(value = "id") Long id) {
        cartItemRepository.deleteById(id);
        return cartItemRepository.getByUserId(userId);

    }

}
```

Figure 12.Cart item controller code



Cart Items

Item Name	Price	Quantity	Action
Gas Burner	1500	1	Delete

Summary

Total Items	1
Total Cost	1500$

Checkout

Figure 13.View of cart

- After reviewing the cart item there is an option for checkout .
- Figure 14 shows As soon as the user clicks on checkout the payment page will appear in this project I have implement typed payment system (dummy) but [as user provide the card details he will get the message on screen successfully placed the order and he will get random order number on screen] card details and order number will be stored in database described in figure 15
- But in terms of cash checkout option then he will get successfully placed the order and he will get random order number on screen

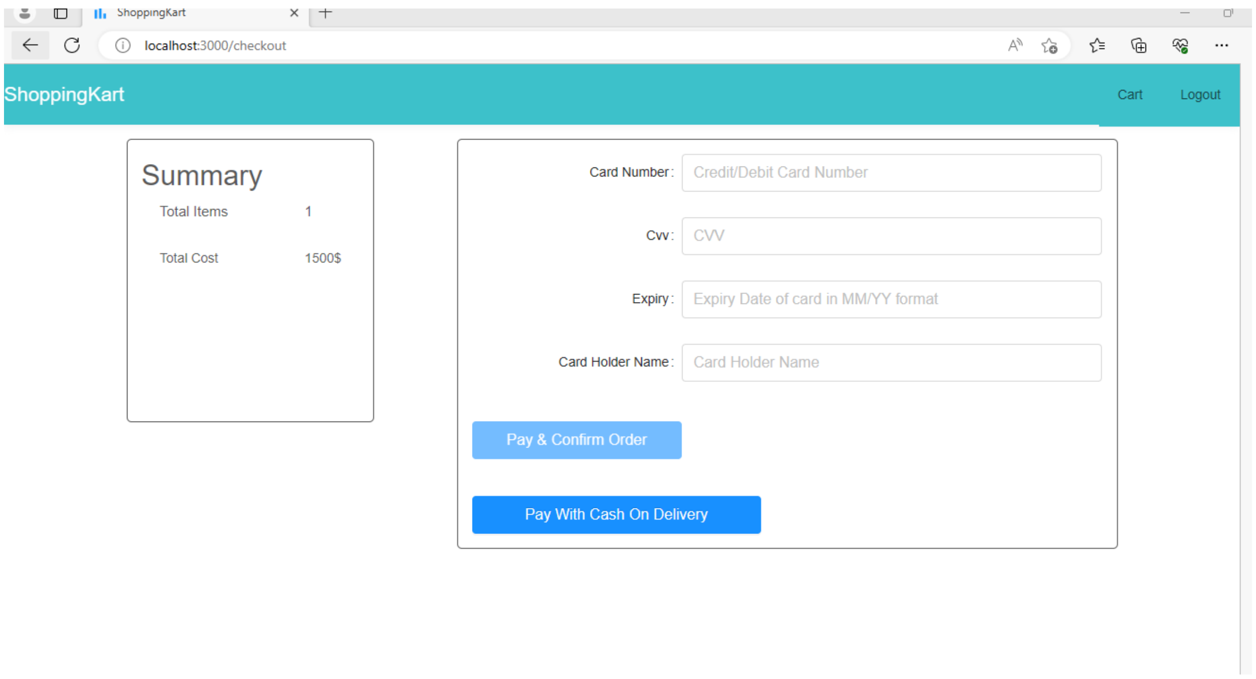
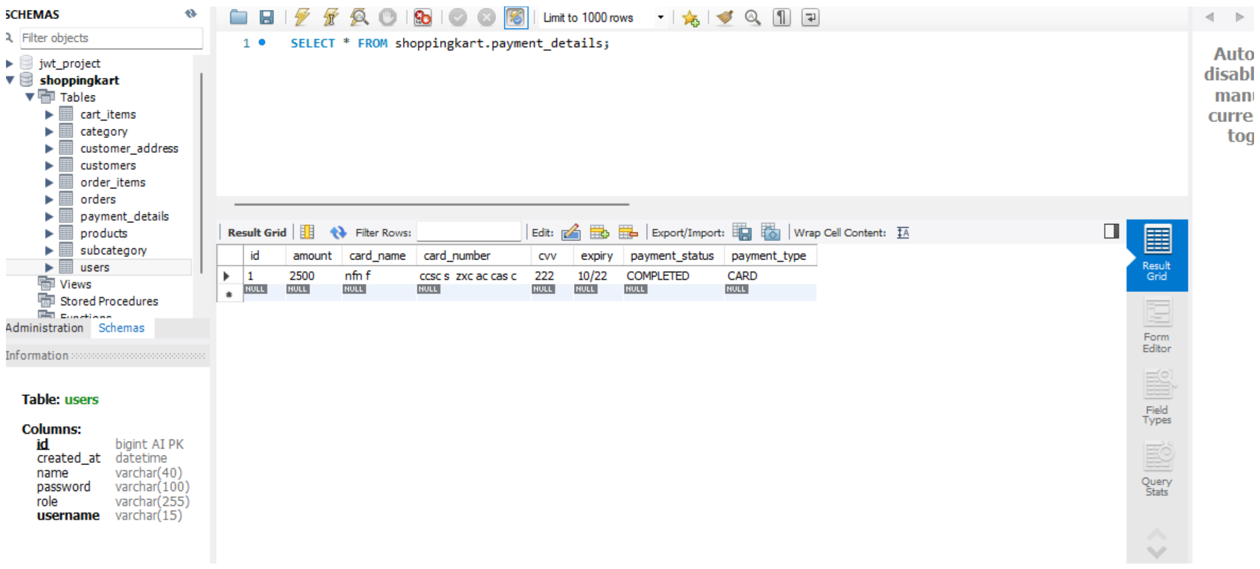

ShoppingKart
localhost:3000/checkout
ShoppingKart
Cart
Logout
Summary
Total Items 1
Total Cost 1500$
Card Number : Credit/Debit Card Number
Cvv : CVV
Expiry : Expiry Date of card in MM/YY format
Card Holder Name : Card Holder Name
Pay & Confirm Order
Pay With Cash On Delivery

Figure 14.View of Ckeckout layout



```sql
SELECT * FROM shoppingkart.payment_details;
```

id	amount	card_name	card_number	cvv	expiry	payment_status	payment_type
1	2500	nfn f	ccsc s zxc ac cas c	222	10/22	COMPLETED	CARD
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

Table: users

Columns:

id	bigint AI PK
created_at	datetime
name	varchar(40)
password	varchar(100)
role	varchar(255)
username	varchar(15)

Figure 15 Database for payment (COD/Card)

3.4 System requirement

- Intel core i3 or above processor
- Memory:- minimum 4gb ram
- Os:- any platform just required the internet
- Tools and framework used:- React,Spring ,Hibernate,API

3.5 software requirement

- Spring STS Eclipse(backend)
- Visual Studio Code(front end)
- Npm(to run react js code)
- Postman(API testing)
- MySQL work bench

- Jkd 11

3.6 UML Diagrams

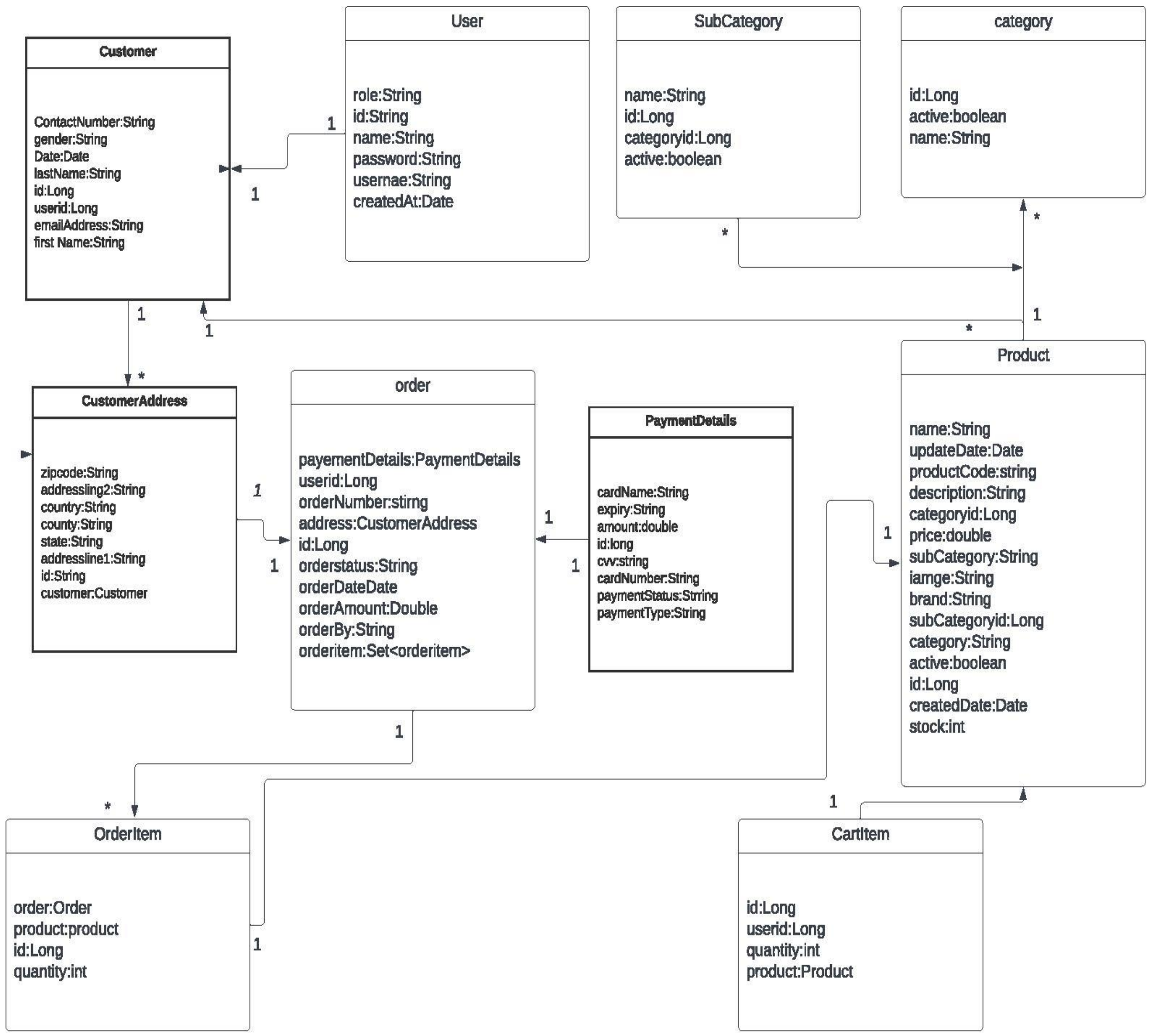
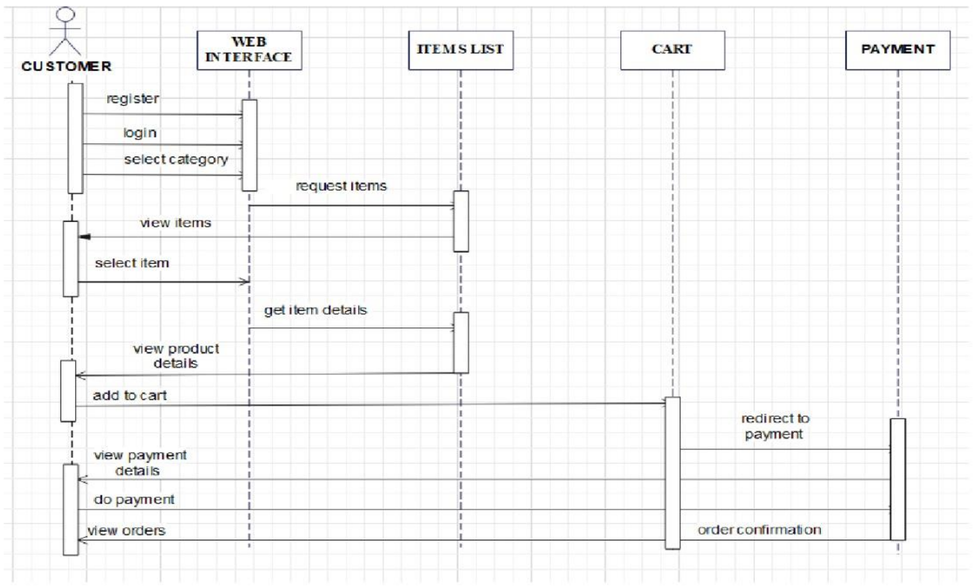


Figure 16. Class diagram

Customer:



Admin:

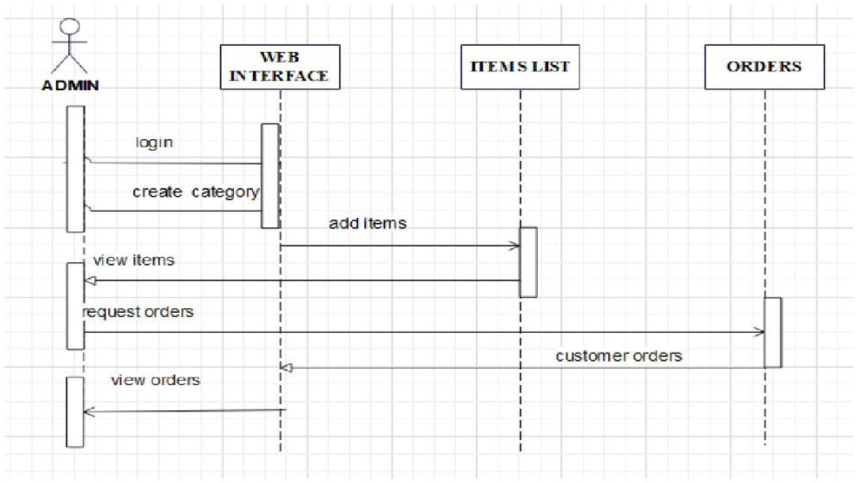


Figure 17.Sequence Diagram

CHAPTER 4 FUTURE WORK

Some potential future work for the development of a full-stack ecommerce web application using Spring Boot, React, Hibernate, and MySQL could include:

1. Integration of a payment gateway: Implementing a secure and reliable payment gateway is crucial for any ecommerce website. Integrating popular payment gateways such as PayPal, Stripe, or Braintree can be a future enhancement.
2. Implementation of search functionality: Enabling customers to search for products quickly and easily can improve the user experience. Implementing a search feature that can search by keywords or product attributes could be a valuable addition.
3. Integration of a recommendation system: Implementing a recommendation system can help customers discover products they may be interested in and improve overall sales. A recommendation system could suggest products based on a customer's purchase history or browsing behavior.
4. Enhanced security features: Security is paramount for any ecommerce website. Additional security features such as two-factor authentication, SSL certificates, and encryption of sensitive data could be implemented to provide customers with a more secure shopping experience.
5. Integration with social media: Integrating social media can help increase brand visibility and connect with customers. Adding social media buttons to product pages or implementing social login options could be a future enhancement.
6. Adding support for multiple languages and currencies: If the ecommerce website is targeting customers in different regions, adding support for multiple languages and currencies can enhance the user experience and increase sales.

7. Implementing order tracking functionality: Providing customers with the ability to track their orders can improve the overall shopping experience. Implementing order tracking functionality can provide customers with real-time updates on the status of their orders.

CHAPTER 5 CONCLUSION

In conclusion, the development of a full-stack ecommerce web application using Spring Boot, React, Hibernate, and MySQL can be a challenging but rewarding project. By utilizing these technologies, it is possible to create a scalable, reliable, and efficient application that can handle large amounts of traffic and data.

Throughout the development process, it is important to pay close attention to the design and architecture of the application, ensuring that it follows best practices and industry standards. Additionally, testing and debugging should be done regularly to ensure that the application is functioning as intended and to catch any potential issues early on.

In the future, there are many potential areas for expansion and improvement, such as adding more features, integrating with third-party APIs, improving the user interface, and optimizing performance. By continuing to iterate and improve the application over time, it can remain competitive and useful for users. Overall, the development of a full-stack ecommerce web application using Spring Boot, React, Hibernate, and MySQL is a valuable and rewarding undertaking for developers and businesses alike.

CHAPTER 6 REFRENCES

1.O.S. Gomez, R.R. Miranda, and K.C. Karen, K.C, “CRUDyLeaf: A DSL for Generating Spring Boot REST APIs from Entity CRUD Operations,” Cybernetics and Information Technologies, vol. 20, pp. 3-14, 2020. DOI: 10.2478/cait-2020-0024

2.An Analysis of the Significance of Spring Boot in The Market M. Mythily; A. Samson Arun Raj; Iwin Thanakumar Joseph 2022 International Conference on Inventive Computation Technologies (ICICT)

3. A javaScript Library for Building User Interfaces [Online] URL: https://facebook.github.io/React/. Accessed February 5 2017

4. E. Wilde, “Putting Things to REST,” Nov. 2007. [Online]. Available: https://escholarship.org/uc/item/1786t1dm

5. L. Li and W. Chou, “Design and describe REST API without violating REST: A petri net based approach,” in 2011 IEEE International Conference on Web Services. IEEE, Jul. 2011. [Online]. Available: https://doi.org/10.1109/icws.2011.54

6. Integration of RESTFul API to Student Information System for Secured Data Sharing and Single Sign-on Philip Alger M. Serrano; Joseph Jessie S. Oñate 2021 IEEE 13th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM).

7. Christian Bauer, Gavin King, and Gary Gregory, Java Persistence with Hibernate, Second Edition. Manning, 2015

8. http://www.devarticles.com.

9. Gupta P, Govil MC (2010) MVC Design pattern for the multi framework distributed applications using XML, spring and struts framework. International Journal on Computer Science and Engineering 2(4): 1047- 1051.

10. Shu-qiang H, Huan-ming Z (2008) Research on improved MVC design pattern based on struts and XSL. 2008 International Symposium on Information Science and Engineering, pp. 451-455.

11. Selfa DM, Carrillo M, Boone MDR (2006) A database and web application based on MVC architecture. 16th International Conference on Electronics, Communications and Computers (CONIELECOMP'06), p. 48.

12.Some information from google.

CHAPTER 7 APPENDIX

BELOW MENTIONED CODE IS FOR PORDUCT WHERE THERE IS BACKEND PART(MODEL,CONTROLLER AND DATABASE) AND FRONT END PART (VIEW)

7.1 MODEL CLASS FOR PRODUCT

```
package com.dev.onlineshopping.model;

import java.util.Date;

import javax.persistence.Column;
import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;
import javax.persistence.Lob;
import javax.persistence.Table;

@Entity
```

```
@Table(name = "products")
public class Product {

	@Id
	@GeneratedValue
	private Long id;
	private String productCode;
	private String name;
	private String description;
	private String category;
	private Long categoryId;
	private String subCategory;
	private Long subCategoryId;
	private double price;
	private String brand;
	private int stock;
	@Column(name = "image")
	@Lob
	private String image;
	private boolean active;
	private Date createdDate;
	private Date updatedDate;

	public Product() {
		super();
	}

	public Product(Long id, String productCode, String name, String description,
String category, Long categoryId,
			String subCategory, Long subCategoryId, double price, String
brand, int stock, String image,
			boolean active) {
		super();
		this.id = id;
		this.productCode = productCode;
		this.name = name;
		this.description = description;
		this.category = category;
		this.categoryId = categoryId;
		this.subCategory = subCategory;
		this.subCategoryId = subCategoryId;
		this.price = price;
		this.brand = brand;
		this.stock = stock;
		this.image = image;
		this.active = active;
}

public Long getId() {
	return id;
}

public void setId(Long id) {
	this.id = id;
}

public String getProductCode() {
	return productCode;
}

public void setProductCode(String productCode) {
	this.productCode = productCode;
}

public String getName() {
	return name;
}

public void setName(String name) {
	this.name = name;
}

public String getDescription() {
	return description;
}

public void setDescription(String description) {
	this.description = description;
}

public String getCategory() {
	return category;
}

public void setCategory(String category) {
	this.category = category;
}

public Long getCategoryId() {
	return categoryId;
}
```

```
public void setCategoryId(Long categoryId) {
        this.categoryId = categoryId;
}

public String getSubCategory() {
        return subCategory;
}

public void setSubCategory(String subCategory) {
        this.subCategory = subCategory;
}

public Long getSubCategoryId() {
        return subCategoryId;
}

public void setSubCategoryId(Long subCategoryId) {
        this.subCategoryId = subCategoryId;
}

public double getPrice() {
        return price;
}

public void setPrice(double price) {
        this.price = price;
}

public String getBrand() {
        return brand;
}

public void setBrand(String brand) {
        this.brand = brand;
}

public int getStock() {
        return stock;
}

public void setStock(int stock) {
        this.stock = stock;
}

public String getImage() {
        return image;
```

```
	}

	public void setImage(String image) {
		this.image = image;
	}

	public boolean isActive() {
		return active;
	}

	public void setActive(boolean active) {
		this.active = active;
	}

	public Date getCreatedDate() {
		return createdDate;
	}

	public void setCreatedDate(Date createdDate) {
		this.createdDate = createdDate;
	}

	public Date getUpdatedDate() {
		return updatedDate;
	}

	public void setUpdatedDate(Date updatedDate) {
		this.updatedDate = updatedDate;
	}

}
```

7.2 CONTROLLER FOR PRODUCT

```
package com.dev.onlineshopping.controller;

import java.util.Date;
import java.util.List;
import java.util.Optional;

import javax.websocket.server.PathParam;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.web.bind.annotation.GetMapping;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.PutMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RestController;

import com.dev.onlineshopping.model.Product;
import com.dev.onlineshopping.repository.ProductRespository;
import com.dev.onlineshopping.requestresponse.ProductRequest;

@RestController
@RequestMapping("/api")
public class ProductController {

	@Autowired ProductRespository productRespository;

	@GetMapping("/product/all")
	List<Product> getAllProducts(){
		return productRespository.findAll();

	}
	@GetMapping("/product/subcategory/{id}")
	List<Product> getAllProducts(@PathVariable(value = "id") Long id){
		return productRespository.findAllBySubCategoryId(id);

	}
	@GetMapping("/product/{id}")
	Optional<Product> getProduct(@PathVariable(value = "id") Long id){
		return productRespository.findById(id);

	}

	@PostMapping("/product")
	Product addProduct(@RequestBody ProductRequest productRequest){
		Product product = new Product(null, productRequest.getProductCode(),
productRequest.getProductName(), productRequest.getDescription(),

	productRequest.getCategoryName(),productRequest.getCategoryId(),
productRequest.getSubCategoryName(),productRequest.getSubCategoryId(),
			productRequest.getPrice(), productRequest.getBrand(),
productRequest.getStock(), productRequest.getImage(), true);
		product.setCreatedDate(new Date());
		product.setUpdatedDate(new Date());
		return productRespository.save(product);

	}
```

```
        @PutMapping("/product")
        Product updateProduct(@RequestBody Product product){
                return productRespository.save(product);

        }

}
```

7.3 API CALL FOR PRODUCT(CONNECTIVITY OF BACKEND TO FRONT END)

```
export function getAllProducts() {
   return request({
      url: API_BASE_URL + "/product/all",
      method: 'GET'
   });
}
```

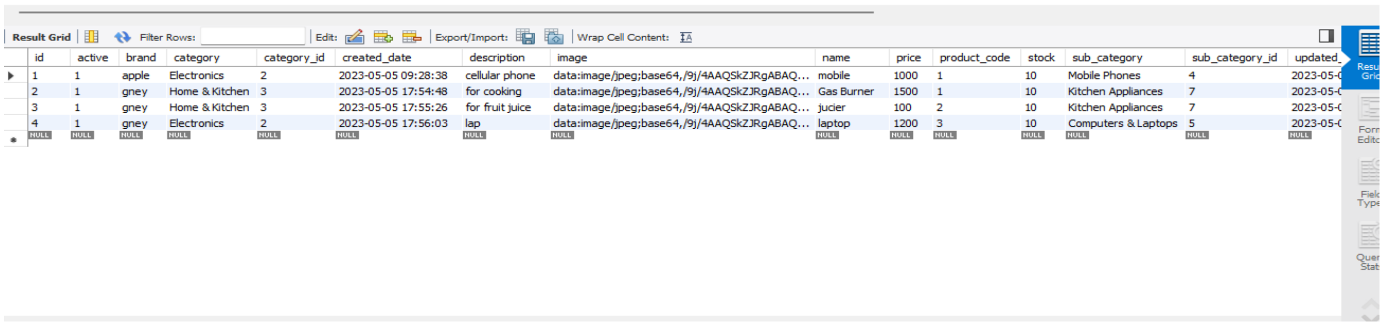
7.4 VIEW CODE FOR PRODUCT

```
import React, { Component } from 'react';
import './Product.css';
import {Modal, Table } from 'antd';
import { deleteProduct } from '../util/APIUtils';
const { confirm } = Modal;

class Product extends Component {
  constructor(props) {
   super(props);
   this.state = {
      products: []
   };
   this.showConfirm = this.showConfirm.bind(this);
}

   showConfirm(id,productId) {
```

```
 confirm({
  title: 'Do you want to delete product?',
  content: 'When clicked the OK button, this product ( ' + productId+ ' ) will get deleted',
  onOk() {
   let promise;
   promise = deleteProduct(id);
   if(!promise) {
    console.log(promise)
     return;
   }
   promise
   .then(response => {
     console.log(response);
     alert("product deleted sucessfully");
     window.location ="/";
   }).catch(error => {
    console.log(error)
   });
  },
  onCancel() {},
 });
}

render() {
  const columns = [
   {
    title: 'Product Code',
    dataIndex: 'productCode',
    key: 'productCode',
   },{
     title: 'product Name',
     dataIndex: 'name',
     key: 'name',
    },
    {
     title: 'Description',
     dataIndex: 'description',
     key: 'description',
    },
    {
     title: 'Category',
     dataIndex: 'category',
     key: 'category',
    },
    {
          title: 'Subcategory',
          dataIndex: 'subCategory',
          key: 'subCategory',
        },
        {
          title: 'Price',
          dataIndex: 'price',
          key: 'price',
        },{
          title: 'Brand',
          dataIndex: 'brand',
          key: 'brand',
        },
        {
          title: 'Stock',
          dataIndex: 'stock',
          key: 'stock',
        // },
        // {
        //   title: 'Action',
        //   dataIndex: '',
        //   key: 'id',
        //   render: (text,record) => <Button onClick={() => this.showConfirm(record.id,record.name)}
size="small" style={{ width: 90 }}>
        //   Delete
        // </Button>,
        }
      ];

     return (
      <div>

        <Table columns={columns} dataSource={this.props.product} />

      </div>

     );
  }
}

export default Product;
```

7.5 PRODUCT DATABASE

```sql
SELECT * FROM shoppingkart.products;
```



id	active	brand	category	category_id	created_date	description	image	name	price	product_code	stock	sub_category	sub_category_id	updated_
1	1	apple	Electronics	2	2023-05-05 09:28:38	cellular phone	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...	mobile	1000	1	10	Mobile Phones	4	2023-05-0
2	1	gney	Home & Kitchen	3	2023-05-05 17:54:48	for cooking	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...	Gas Burner	1500	1	10	Kitchen Appliances	7	2023-05-0
3	1	gney	Home & Kitchen	3	2023-05-05 17:55:26	for fruit juice	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...	jucier	100	2	10	Kitchen Appliances	7	2023-05-0
4	1	gney	Electronics	2	2023-05-05 17:56:03	lap	data:image/jpeg;base64,/9j/4AAQSkZJRgABAQ...	laptop	1200	3	10	Computers & Laptops	5	2023-05-0
NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL