A Report on Capstone Project

SHOP FOR HOME

done by

Lakshmi Vineetha

Bhavya Kancharla

Sumanth Kakileti

Mahima Kumari

Daneswari Aketi

of Batch -**Wipro Velocity AWS_Apr CIII** -
Group-13

Under the guidance of

Praveen Tej

Abstract

The Shop For Home is an E-Commerce Website which is developed for shopping required products like groceries, home appliances, electronic devices, mobiles, beauty items through online without direct interaction of the customer to the seller. The project is developed by using the technologies Angular, Springboot and MySQL. The front-End is developed by using Angular, the backend is developed by using Springboot and MySQL is the database which is used for data storing.

Online platform provides convenience for customers as they do not have to leave home and only need to browse website online, especially for buying the products which are not sold in nearby shops. It could help customers buy wider range of products and save customers time. Consumers also gain product information through online shopping and research products and compare prices among retailers.

Table of Content

Introduction

E-Commerce stands for electronic commerce, as in online. It's an umbrella term for any transaction done over the internet. E-Commerce includes retail stores, such as clothing and other physical products, and services of all types, from cyber security to booking a hotel. The most common way to facilitate online sales and transactions, is through a dedicated online store.

E-commerce creates new job opportunities based on information related services, software app and digital products. It also causes job losses like areas with the greatest predicted job-loss are retail, postal, and travel agencies. The development of e-commerce will create jobs that require highly skilled workers to manage large amounts of information, customer demands, and production processes. In contrast, people with poor technical skills cannot enjoy the wages welfare. On the other hand, because e-commerce requires sufficient stocks that could be delivered to customers in time, the warehouse becomes an important element. Warehouse needs more staff to manage, supervise and organize, thus the condition of warehouse environment will be concerned by employees.

However, e-commerce lacks human interaction for customers, especially who prefer face-to-face connection. Customers are also concerned with the security of online transactions and tend to remain loyal to well-known retailers. In recent years, clothing retailers such as Tommy

Hilfiger have started adding Virtual Fit platforms to their e-commerce sites to reduce the risk of customers buying the wrong sized clothes, although these vary greatly in their fit for purpose. When the customer regret the purchase of a product, it involves returning goods and refunding process. This process is inconvenient as customers need to pack and post the goods. If the products are expensive, large or fragile, it refers to safety issues.

E-commerce brings convenience for customers as they do not have to leave home and only need to browse website online, especially for buying the products which are not sold in nearby shops. It could help customers buy wider range of products and save customers' time. Consumers also gain power through online shopping. They research products and compare prices among retailers. Also, online shopping often provides sales promotion or discounts code, thus it is more price effective for customers. Moreover, e-commerce provides products' detailed information; even the in-store staff cannot offer such detailed explanation. Customers can also review and track the order history online.

Technologies Used:

Angular:

Angular is a platform and framework for building single-page client applications using HTML and TypeScript. Angular was written in TypeScript. It implements core and optional functionality as a set of

TypeScript libraries that you import into applications.

The basic building blocks of the Angular framework are Angular components that are organized into Ng Modules.

- Ng Modules collect related code into functional sets; an Angular application is defined by a set of Ng Modules.
- An application always has at least a root module that enables bootstrapping, and typically has many more feature modules.

Components are the building blocks that compose an application. A component includes a TypeScript class with a @Component() decorator, an HTML template, and styles. The @Component() decorator specifies the following Angular-specific information:

- A CSS selector that defines how the component is used in a template. HTML elements in your template that match this selector become instances of the component.
- An HTML template that instructs Angular how to render the component
- An optional set of CSS styles that define the appearance of the template's HTML elements

Modules, components and services are classes that use decorators. These decorators mark their type and provide metadata that tells Angular how to use them.

- The metadata for a component class associates it with a *template* that defines a view. A template combines ordinary HTML with Angular *directives* and *binding markup* that allow Angular to modify the HTML before rendering it for display.
- The metadata for a service class provides the information Angular needs to make it available to components through dependency injection (DI).

Angular provides the Router service to help you define navigation paths among views. The router provides sophisticated in-browser navigational capabilities.

Springboot:

Spring Boot makes it easy to create stand-alone, production-grade Spring based Applications that can be just run applications.

We take an opinionated view of the Spring platform and third-party libraries so you can get started with minimum fuss. Most Spring Boot applications need minimal Spring configuration.

Features

- Create stand-alone Spring applications
- Embed Tomcat, Jetty or Undertow directly (no need to deploy WAR files)

- Provide opinionated 'starter' dependencies to simplify your build configuration
- Automatically configure Spring and 3rd party libraries whenever possible
- Provide production-ready features such as metrics, health checks, and externalized configuration.

Java Spring Boot is a tool that makes developing web application and microservices ,by Spring Framework faster and easier through three core capabilities:

1. Autoconfiguration
2. An opinionated approach to configuration
3. The ability to create standalone applications

These features work together to provide you with a tool that allows you to set up a Spring-based applicationwith minimal configuration and setup. Spring is widely used for creating scalable applications. For web applications Spring provides Spring MVC which is a widely used module of spring which is used to create scalable web applications.

But main disadvantage of spring projects is that configuration is really time-consume and can be a bit overwhelming for the new developers. Making the application production-ready takes some time if you are new to the spring.

Solution to this is Spring Boot. Spring Boot is built on the top of the spring and contains all the features of spring. And is becoming favourite of developer's these days because of it's a rapid production-ready environment which enables the developers to directly focus on the logic instead of struggling with the configuration and set up.

Spring Boot is a microservice-based framework and making a production-ready application in it takes very less time. Prerequisite for Spring Boot is the basic knowledge Spring framework.

MySQL:

MySQL is a fast, easy-to-use RDBMS being used for many small and big businesses. MySQL is developed, marketed and supported by MySQL AB, which is a Swedish company. MySQL is becoming so popular because of many good reasons -

- MySQL is released under an open-source license. So you have nothing to pay to use it.
- MySQL is a very powerful program in its own right. It handles a large subset of the functionality of the most expensive and powerful database packages.
- MySQL uses a standard form of the well-known SQL data language.
- MySQL works on many operating systems and with many languages including PHP, PERL, C, C++, JAVA, etc.
- MySQL works very quickly and works well even with large data sets.

- MySQL is very friendly to PHP, the most appreciated language for web development.
- MySQL supports large databases, up to 50 million rows or more in a table. The default file size limit for a table is 4GB, but you can increase this (if your operating system can handle it) to a theoretical limit of 8 million terabytes (TB).
- MySQL is customizable. The open-source GPL license allows programmers to modify the MySQL software to fit their own specific environments.

Problem Statement:

ShopForHome is a popular Store in the market for shopping the home décor stuff. Due to Covid 19 all the offline shopping stopped. So, the store wants to move to the online platforms and wants their own web application.

There are 2 users
on the application: -
1.User
2.Admin

User Stories –

1.As a user I should be able to login, Logout and
Register into the application.2.As a user I should
be able to see the products in different categories.
3.As a user I should be able to sort the products.

4.As a user I should be able to add the products into the shopping cart.
5.As a user I should be able to increase or decrease
the quantity added in the cart.6.As a user I should
be able to add “n” number of products in the cart.
7.As a user I should be able to get the Wishlist option where I can add those products which I want but don’twant to order now.
8.As a user I should get different discount coupons.

Admin Stories –

1.As an Admin I should be able to login, Logout and
Register into the application.2.As an Admin I should
be able to perform CRUD on Users.
3.As an Admin I should be able to Perform CRUD on the products.
4.As an Admin I should be able to get bulk upload option to
upload a csv for products details.5.As an Admin I should be
able to get the stocks.
6.As an Admin I should be able to mail if any stock is less than 10.
7.As an Admin I should be able to get the sales report of a specific duration.
8.As an Admin I should be able to set the discount coupons for the specific set of users.

Instructions –

1. Please use a folder on server to upload the images.
2. Please share the database structure in the .sql file.
3. Please create a separate microservice for reports and discount coupons.
4. Please use separate port to deploy the Angular UI and Spring Boot Microservice.
5. Please use the UI designing tool like (Bootstrap or Material) to make your UI better.
6. Please use Material UI to create the UI.

Frontend Implementation

Initially Install the Angular within the PC by using the command npm install -g @angular/cli

After installation,

1.Created a Repository:

Created a repository in Git-Hub to export the source code and share within the team.

2.Creating Project:

Create an angular project by using the command **ng new Project-Name** (Shop from home).

The project is generated after the installation of required package. Open the project in the Visual Studio Code.

After opening the project with the VS code create the components.

3.Creating Components:

For creating the components use the command **ng generate component component-name**.

The Components within the project are:

1. admin

2. adminlogin

3. cart

4. checkout

5. discount

6.e-commerce

7. header

8. payment

9. products

10. uploadfiles

11. userlogin

12. userregister

13. users

14. wishlist

15.service

16.shared.

4.Creating the static template of e-commerce website:

By using angular in vs code we developed the structured code using HTML, adds styling using CSS, and adds the backend actions by using java script.

The structured code is stored on html page, the css will be stored in the .css file and the actions will be stored in the .ts file.

In this template we created the all pages like homepage, cart page, sign in and signup page, wishlist page, and the categories page in the homepage.

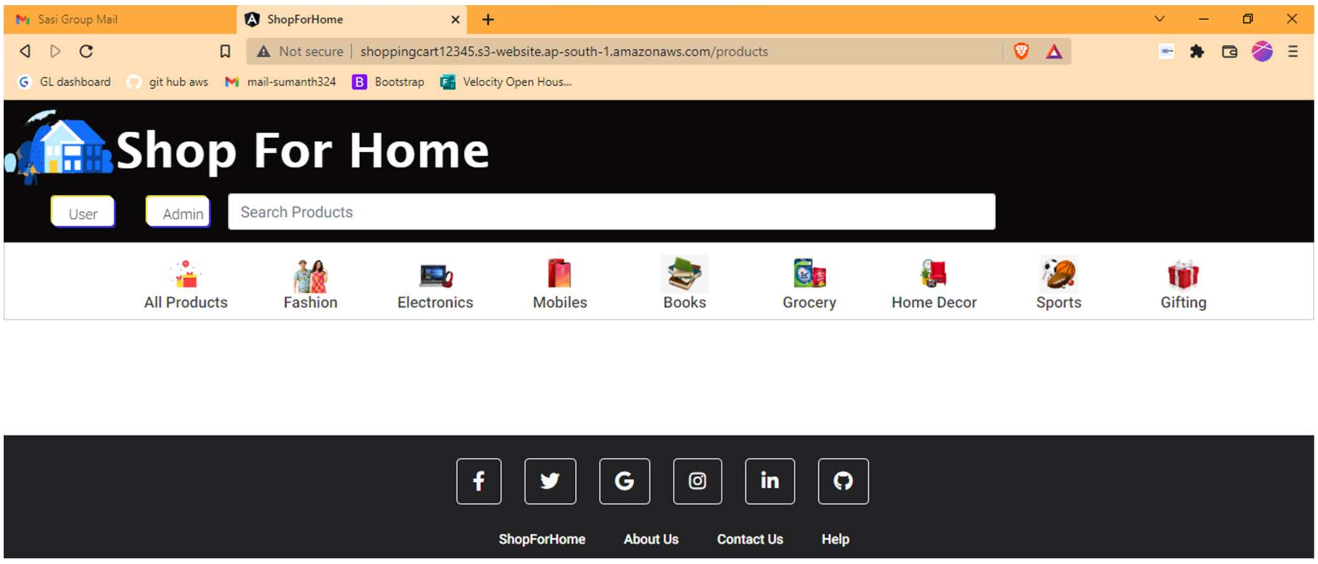


Fig- Homepage

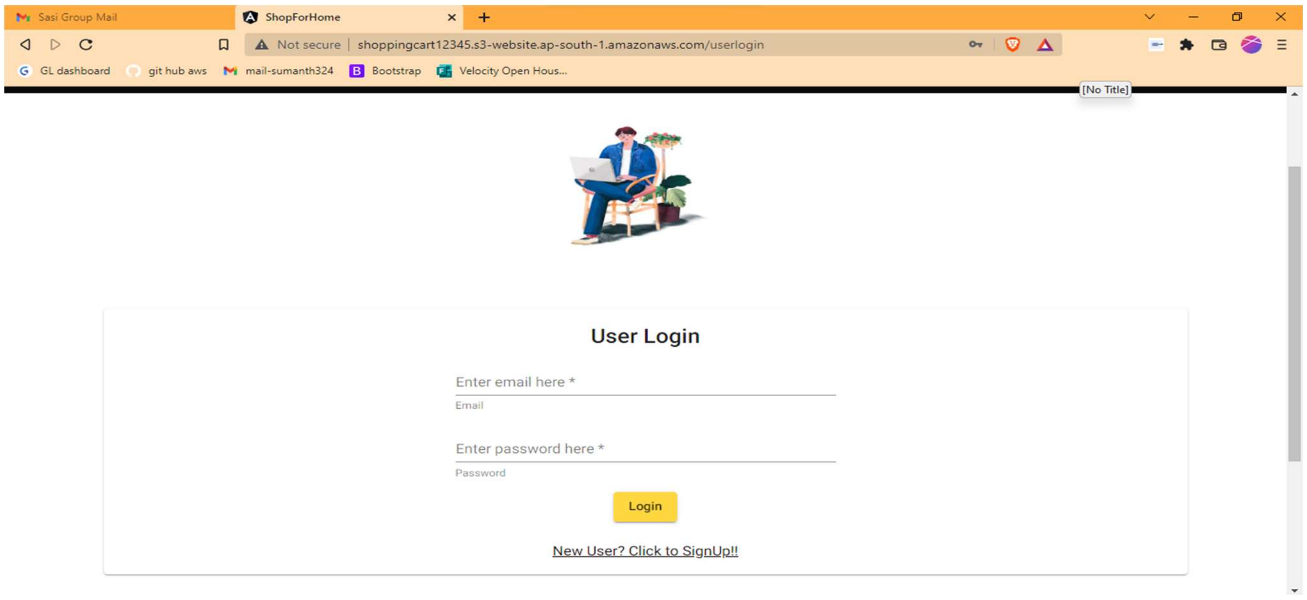


Fig- Sign-in page

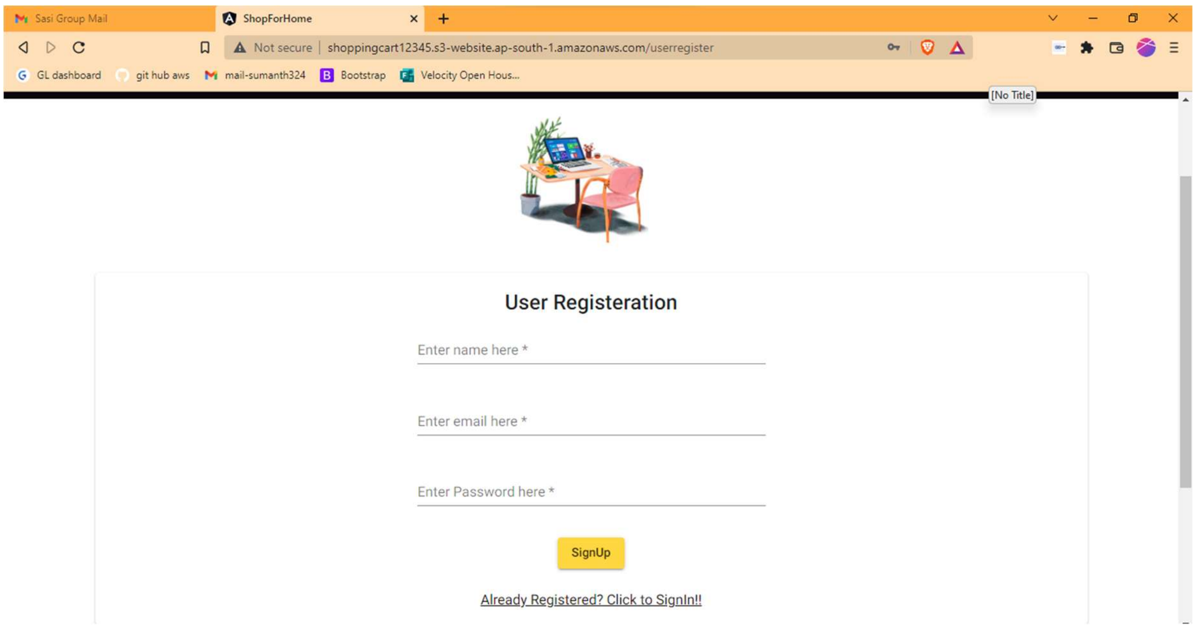
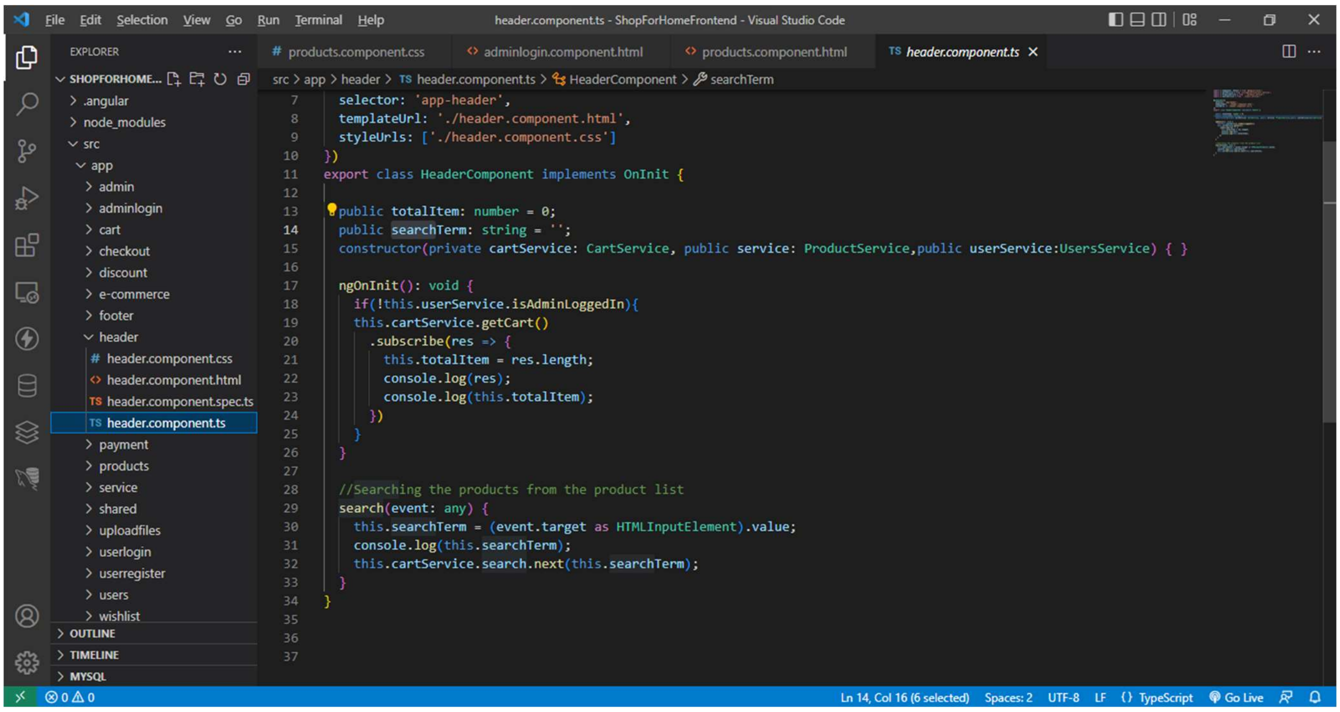


Fig – Sign-up page

5. Creates function search:

By using type script and html creates the search function by comparing the user input in the data of products we have in the data base.



```typescript
  selector: 'app-header',
  templateUrl: './header.component.html',
  styleUrls: ['./header.component.css']
})
export class HeaderComponent implements OnInit {

  public totalItem: number = 0;
  public searchTerm: string = '';
  constructor(private cartService: CartService, public service: ProductService,public userService:UsersService) { }

  ngOnInit(): void {
    if(!this.userService.isAdminLoggedIn){
    this.cartService.getCart()
      .subscribe(res => {
        this.totalItem = res.length;
        console.log(res);
        console.log(this.totalItem);
      })
    }
  }

  //Searching the products from the product list
  search(event: any) {
    this.searchTerm = (event.target as HTMLInputElement).value;
    console.log(this.searchTerm);
    this.cartService.search.next(this.searchTerm);
  }
}
```

fig- search function

6. Creating categories:

By using the data base while inserting the product data we included the category id by that we created the categories in the homepage. If the select single category it will show on the products in the category id.

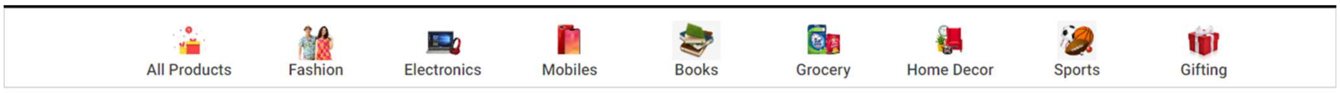
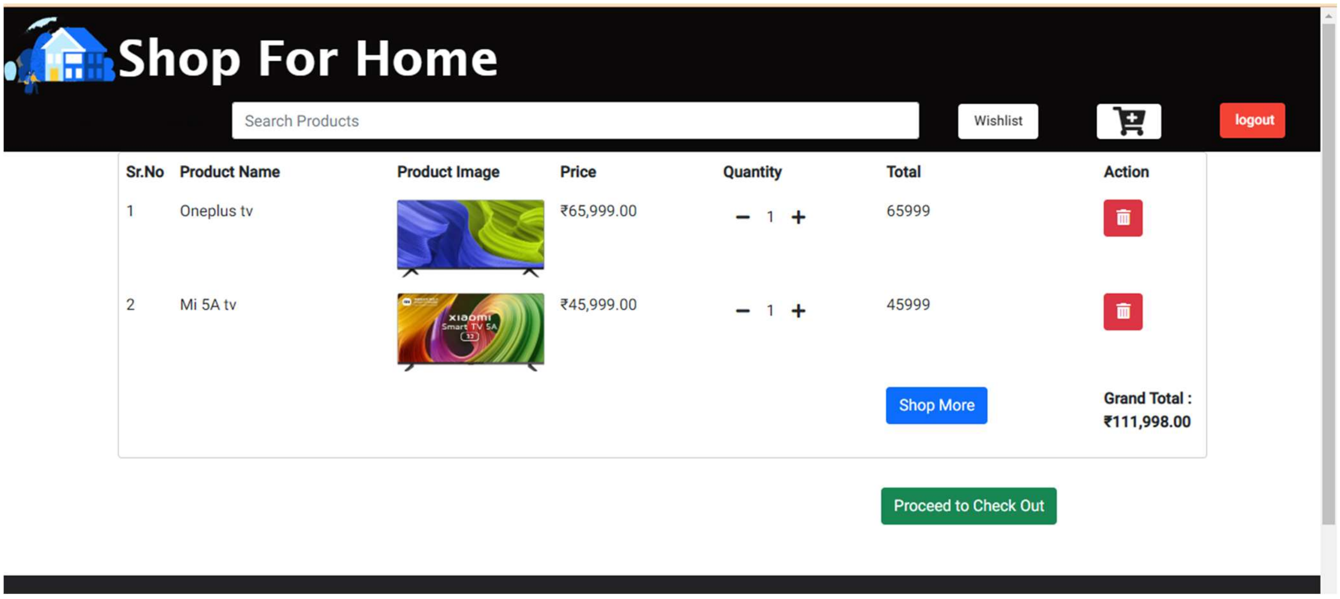


Fig- catagories

7. Creating cart:

In the card module the cart.html had the structure of cart page, in css we added the cart styling and in the ts file we had the actions like adding and removing by adding the product in the cart table in database and removing from database.



Sr.No	Product Name	Product Image	Price	Quantity	Total	Action
1	Oneplus tv		₹65,999.00	− 1 +	65999	
2	Mi 5A tv		₹45,999.00	− 1 +	45999	

Grand Total : ₹111,998.00

Fig- cart

8.Execution:

Make sure that app-routing.module.ts is well written and importing of respective component modules are done perfectly.

After the completing of implementation part for executing the project by using the command **ng serve**

If the code is error free it executes perfectly by showing the port number.

Backend Implementation

1.Initialization of Spring Boot:

To start with Spring Boot REST API, you first need to initialize the Spring Boot Project. You caneasily initialize a new Spring Boot Project with Spring Initializer.

From your Web Browser, go to start.spring.io. Choose Maven as your Build Tool and Language as Java.Select the specific version of Spring Boot you want to go ahead with.

Add dependencies:

1. Spring Web
2. Spring Boot DevTools
3. Spring Data JPA
4. MySQL Driver
5. JDBC API
6. Web Socket

2.Connecting Spring Boot to the Database:

Next, you need to set up the Database, and you can do it easily with Spring Data JPA.

Add some elementary information in your application. Properties file to set up the connection to your preferred Database. Add your JDBC connection URL, provide a username and password for authentication, and set the ddl-auto property to update.

3.Creating Required Packages:

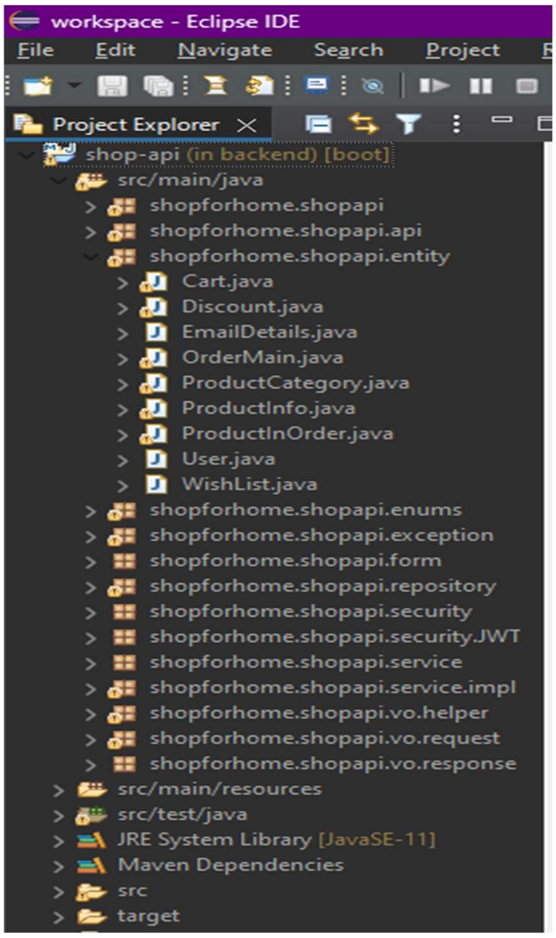
Create new packages as per requirement within the initiated project.

The packages are generated in the project are mentioned below:

1. Config
2. Controller
3. Exception
4. Model
5. Repository
6. Service

4.Creation of entities:

creation of entities in springboot application, as shown in below

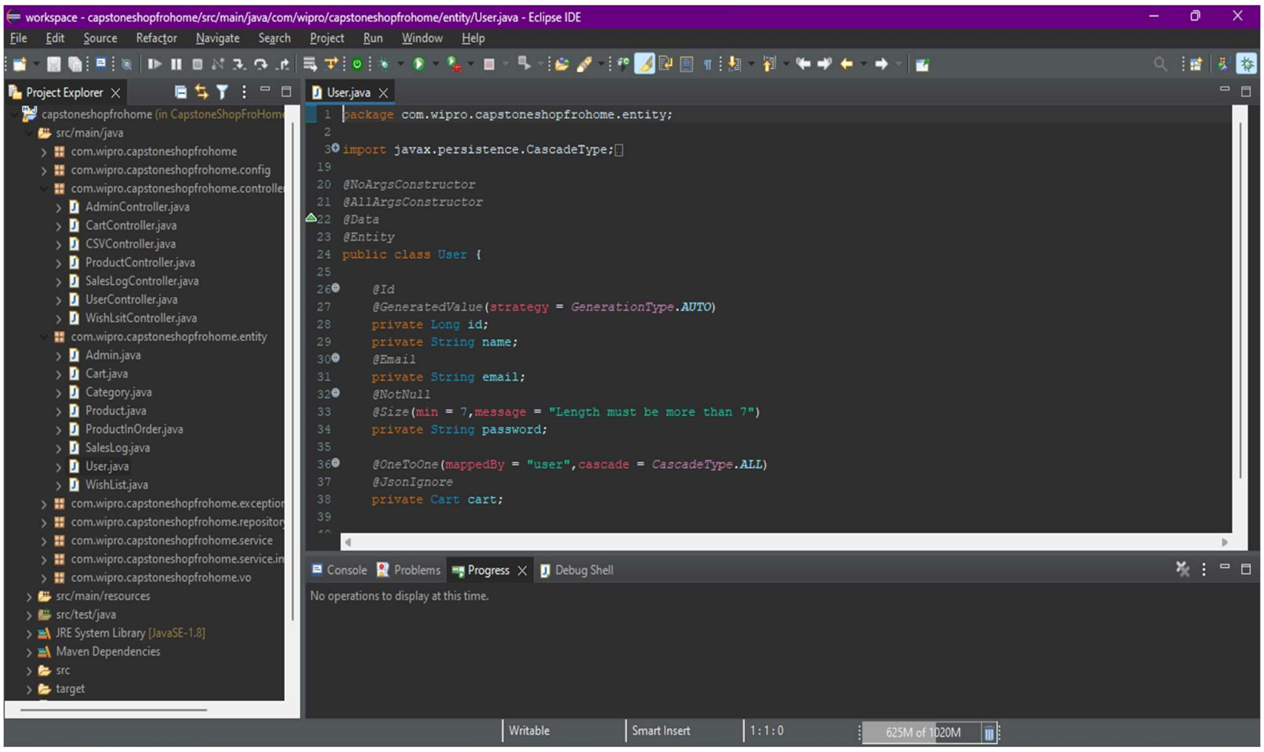


```
workspace - Eclipse IDE
File  Edit  Navigate  Search  Project
Project Explorer
shop-api (in backend) [boot]
  src/main/java
    shopforhome.shopapi
    shopforhome.shopapi.api
    shopforhome.shopapi.entity
      Cart.java
      Discount.java
      EmailDetails.java
      OrderMain.java
      ProductCategory.java
      ProductInfo.java
      ProductInOrder.java
      User.java
      WishList.java
    shopforhome.shopapi.enums
    shopforhome.shopapi.exception
    shopforhome.shopapi.form
    shopforhome.shopapi.repository
    shopforhome.shopapi.security
    shopforhome.shopapi.security.JWT
    shopforhome.shopapi.service
    shopforhome.shopapi.service.impl
    shopforhome.shopapi.vo.helper
    shopforhome.shopapi.vo.request
    shopforhome.shopapi.vo.response
  src/main/resources
  src/test/java
  JRE System Library [JavaSE-11]
  Maven Dependencies
  src
  target
```

Fig- creation of entites

5.Creation of Data Transfer objects:

Creation of Data Transfer objects – boot as shown in below


workspace - capstoneshopfrohome/src/main/java/com/wipro/capstoneshopfrohome/entity/User.java - Eclipse IDE
File Edit Source Refactor Navigate Search Project Run Window Help
Project Explorer
capstoneshopfrohome (in CapstoneShopFroHome
src/main/java
com.wipro.capstoneshopfrohome
com.wipro.capstoneshopfrohome.config
com.wipro.capstoneshopfrohome.controlle
AdminController.java
CartController.java
CSVController.java
ProductController.java
SalesLogController.java
UserController.java
WishLsitController.java
com.wipro.capstoneshopfrohome.entity
Admin.java
Cart.java
Category.java
Product.java
ProductInOrder.java
SalesLog.java
User.java
WishList.java
com.wipro.capstoneshopfrohome.exception
com.wipro.capstoneshopfrohome.repositor
com.wipro.capstoneshopfrohome.service
com.wipro.capstoneshopfrohome.service.in
com.wipro.capstoneshopfrohome.vo
src/main/resources
src/test/java
JRE System Library [JavaSE-1.8]
Maven Dependencies
src
target
User.java
Console Problems Progress Debug Shell
Writable
Smart Insert
1 : 1 : 0
625M of 1020M

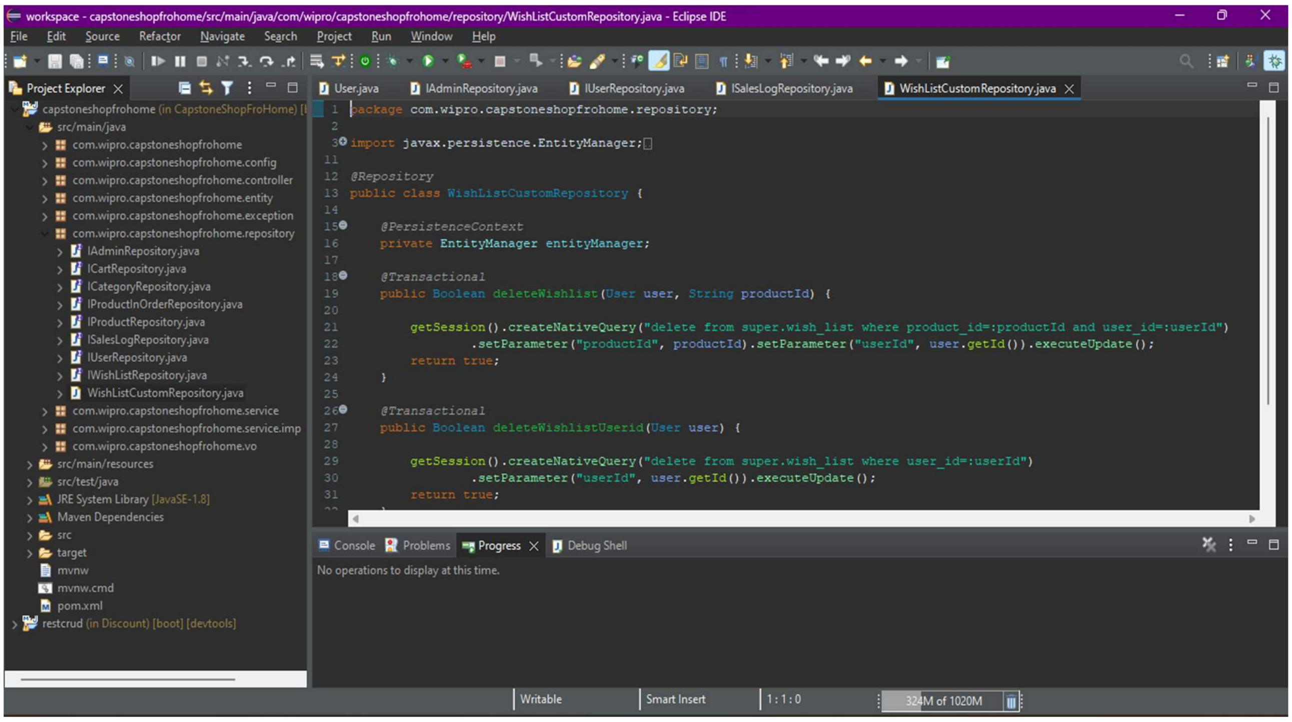
```
package com.wipro.capstoneshopfrohome.entity;

import javax.persistence.CascadeType;

@NoArgsConstructor
@AllArgsConstructor
@Data
@Entity
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.AUTO)
    private Long id;
    private String name;
    @Email
    private String email;
    @NotNull
    @Size(min = 7,message = "Length must be more than 7")
    private String password;

    @OneToOne(mappedBy = "user",cascade = CascadeType.ALL)
    @JsonIgnore
    private Cart cart;

```

```
No operations to display at this time.
```

Fig-data transfer objects

6.Creation repository:

Creation of repository in spring-boot,



workspace - capstoneshopfrohome/src/main/java/com/wipro/capstoneshopfrohome/repository/WishListCustomRepository.java - Eclipse IDE

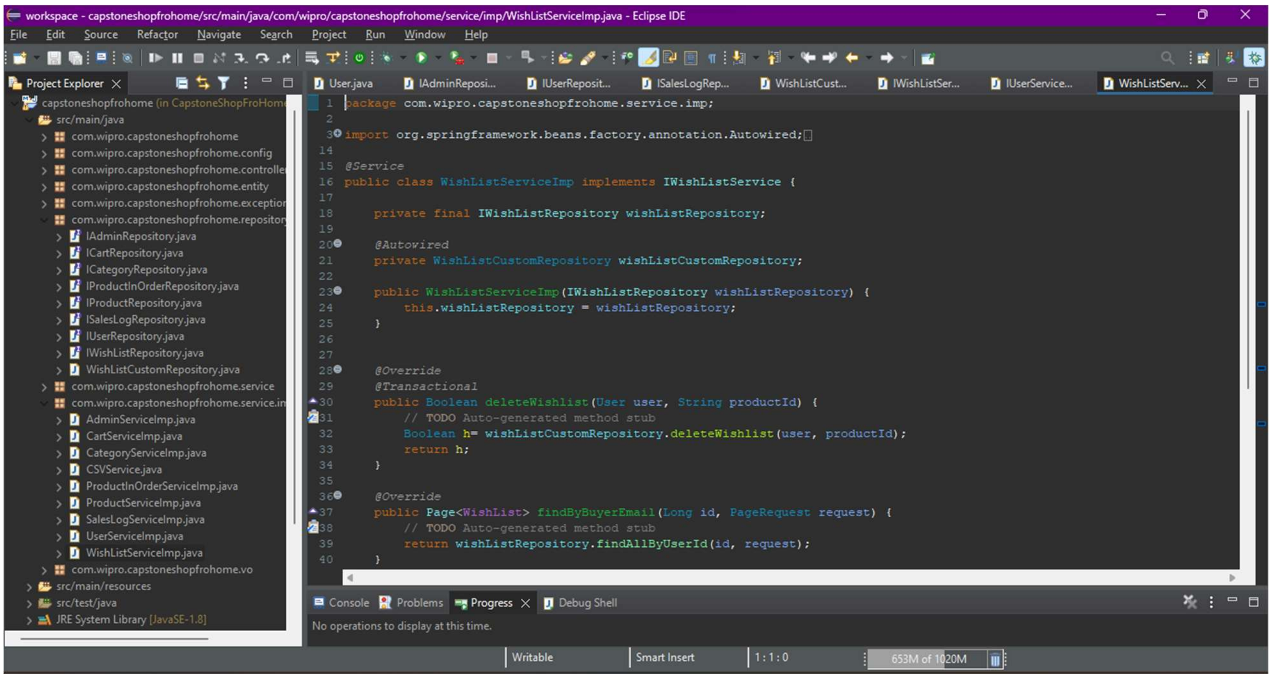
```java
package com.wipro.capstoneshopfrohome.repository;

import javax.persistence.EntityManager;

@Repository
public class WishListCustomRepository {

    @PersistenceContext
    private EntityManager entityManager;

    @Transactional
    public Boolean deleteWishlist(User user, String productId) {

        getSession().createNativeQuery("delete from super.wish_list where product_id=:productId and user_id=:userId")
                .setParameter("productId", productId).setParameter("userId", user.getId()).executeUpdate();
        return true;
    }

    @Transactional
    public Boolean deleteWishlistUserid(User user) {

        getSession().createNativeQuery("delete from super.wish_list where user_id=:userId")
                .setParameter("userId", user.getId()).executeUpdate();
        return true;
```

No operations to display at this time.

fig- repository

7. Create Service layer logic:

Creation of service layer in spring-boot as shown in below,



workspace - capstoneshopfrohome/src/main/java/com/wipro/capstoneshopfrohome/service/imp/WishListServiceImp.java - Eclipse IDE

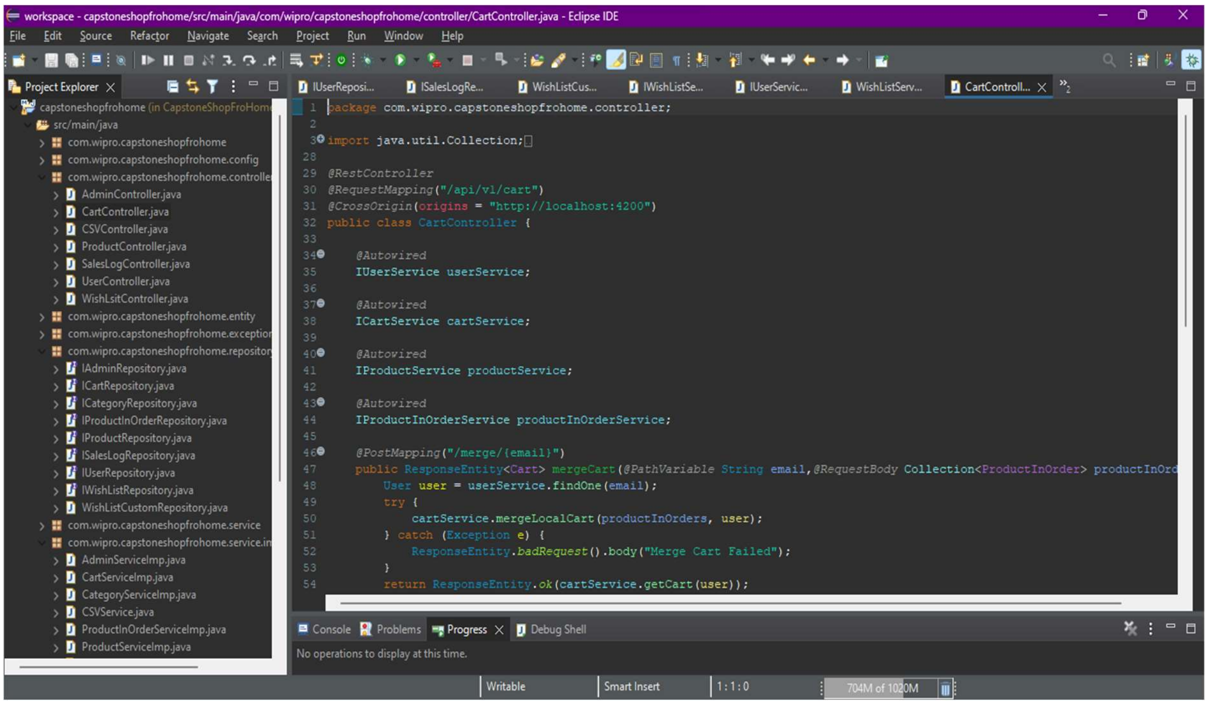
```java
package com.wipro.capstoneshopfrohome.service.imp;

import org.springframework.beans.factory.annotation.Autowired;

@Service
public class WishListServiceImp implements IWishListService {

    private final IWishListRepository wishListRepository;

    @Autowired
    private WishListCustomRepository wishListCustomRepository;

    public WishListServiceImp(IWishListRepository wishListRepository) {
        this.wishListRepository = wishListRepository;
    }


    @Override
    @Transactional
    public Boolean deleteWishlist(User user, String productId) {
        // TODO Auto-generated method stub
        Boolean h= wishListCustomRepository.deleteWishlist(user, productId);
        return h;
    }

    @Override
    public Page<WishList> findByBuyerEmail(Long id, PageRequest request) {
        // TODO Auto-generated method stub
        return wishListRepository.findAllByUserId(id, request);
    }
```

No operations to display at this time.

Fig- service layer

8. **Create Controller to direct rest api**:

Creation of rest api as shown in below,



```java
package com.wipro.capstoneshopfrohome.controller;

import java.util.Collection;

@RestController
@RequestMapping("/api/v1/cart")
@CrossOrigin(origins = "http://localhost:4200")
public class CartController {

    @Autowired
    IUserService userService;

    @Autowired
    ICartService cartService;

    @Autowired
    IProductService productService;

    @Autowired
    IProductInOrderService productInOrderService;

    @PostMapping("/merge/{email}")
    public ResponseEntity<Cart> mergeCart(@PathVariable String email,@RequestBody Collection<ProductInOrder> productInOrd
        User user = userService.findOne(email);
        try {
            cartService.mergeLocalCart(productInOrders, user);
        } catch (Exception e) {
            ResponseEntity.badRequest().body("Merge Cart Failed");
        }
        return ResponseEntity.ok(cartService.getCart(user));
```

Fig- rest api

9.**Creating Discount Microservices:**

By adding the required dependencies and packages, developed the code as per the requirement.

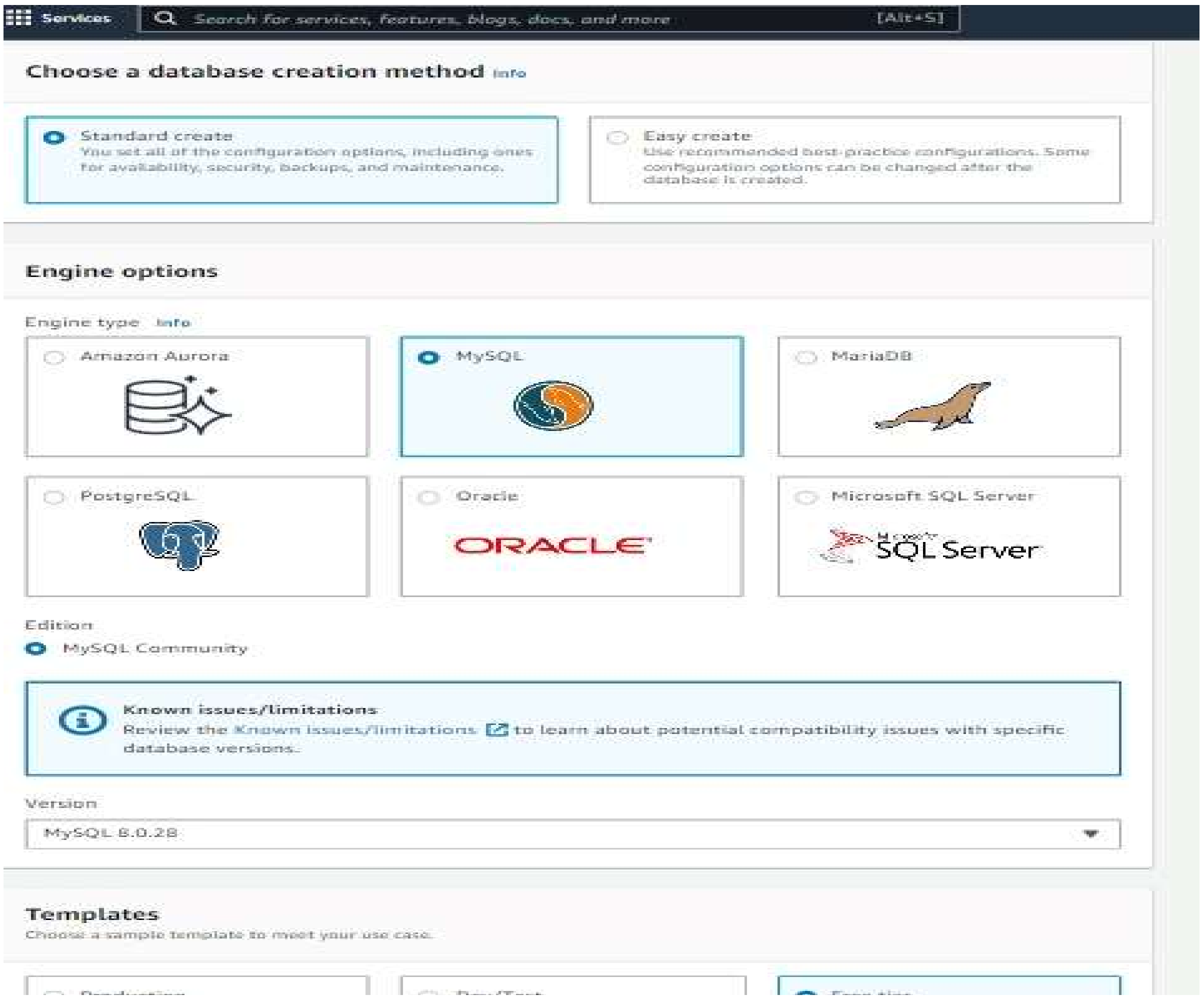
Add some elementary information in your application. Properties file to set up the connection to your preferred Database. Add your JDBC connection URL, provide a username and password for authentication, and set the ddl-auto property to update. And add server port within the application. Properties by using server.

AWS deployment:

By the successful execution of backend developed project with the Spring Tool Suite then there is generationof data tables within the triggered database location.

After creation of data tables within the database location, Add admin details with in the admin tables byusing the database query INSERT, like **insert into admin values('admin@gmail.com','admin').**

1.Choosing A Database: We select database as per our requirement and select template for the database design and add configurations as required.



Choose a database creation method Info

[✓ Standard create] You set all of the configuration options, including ones for availability, security, backups, and maintenance.

Easy create — Use recommended best-practice configurations. Some configuration options can be changed after the database is created.

Engine options

Engine type Info

Amazon Aurora | [✓ MySQL] | MariaDB | PostgreSQL | Oracle | Microsoft SQL Server

Edition

[✓ MySQL Community]

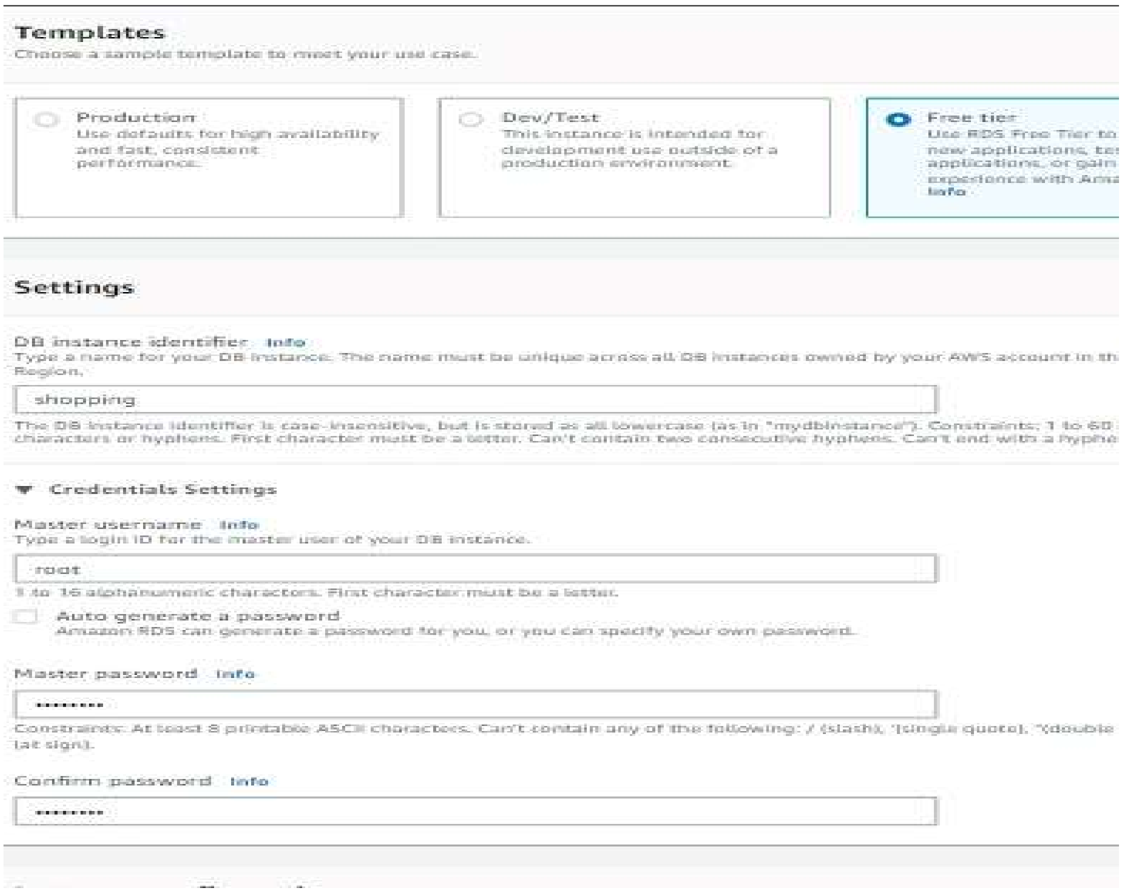
Known issues/limitations
Review the Known issues/limitations to learn about potential compatibility issues with specific database versions.

Version

MySQL 8.0.28

Templates
Choose a sample template to meet your use case.

Fig-creation of database



Templates
Choose a sample template to meet your use case.

( ) Production
Use defaults for high availability and fast, consistent performance.

( ) Dev/Test
This instance is intended for development use outside of a production environment.

(●) Free tier
Use RDS Free Tier to new applications, tes applications, or gain experience with Ama
Info

Settings

DB instance identifier Info
Type a name for your DB instance. The name must be unique across all DB instances owned by your AWS account in th Region.
shopping
The DB instance identifier is case-insensitive, but is stored as all lowercase (as in "mydbinstance"). Constraints: 1 to 60 characters or hyphens. First character must be a letter. Can't contain two consecutive hyphens. Can't end with a hyphe

▼ Credentials Settings

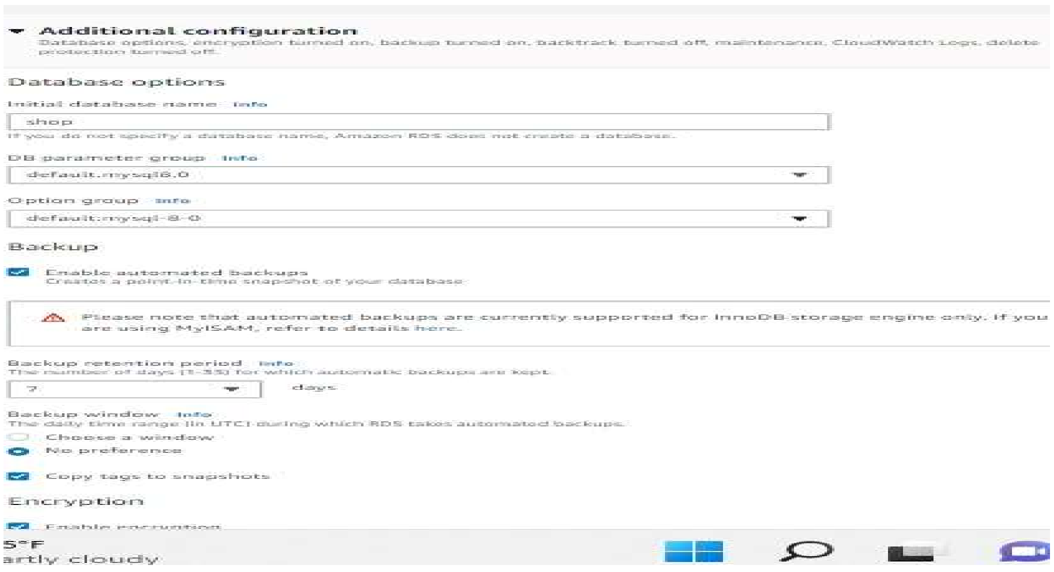
Master username Info
Type a login ID for the master user of your DB instance.
root
1 to 16 alphanumeric characters. First character must be a letter.

☐ Auto generate a password
Amazon RDS can generate a password for you, or you can specify your own password.

Master password Info
•••••••••
Constraints: At least 8 printable ASCII characters. Can't contain any of the following: / (slash), '(single quote), "(double (at sign).

Confirm password Info
•••••••••

Fig-selection of Template



▼ Additional configuration
Database options, encryption turned on, backup turned on, backtrack turned off, maintenance, CloudWatch Logs, delete protection turned off.

Database options

Initial database name Info
shop
If you do not specify a database name, Amazon RDS does not create a database.

DB parameter group Info
default.mysql8.0

Option group Info
default:mysql-8-0

Backup

☑ Enable automated backups
Creates a point-in-time snapshot of your database

Please note that automated backups are currently supported for InnoDB storage engine only. If you are using MyISAM, refer to details here.

Backup retention period Info
The number of days (1-35) for which automatic backups are kept.
7 days

Backup window Info
The daily time range (in UTC) during which RDS takes automated backups.
( ) Choose a window
(●) No preference

☑ Copy tags to snapshots

Encryption

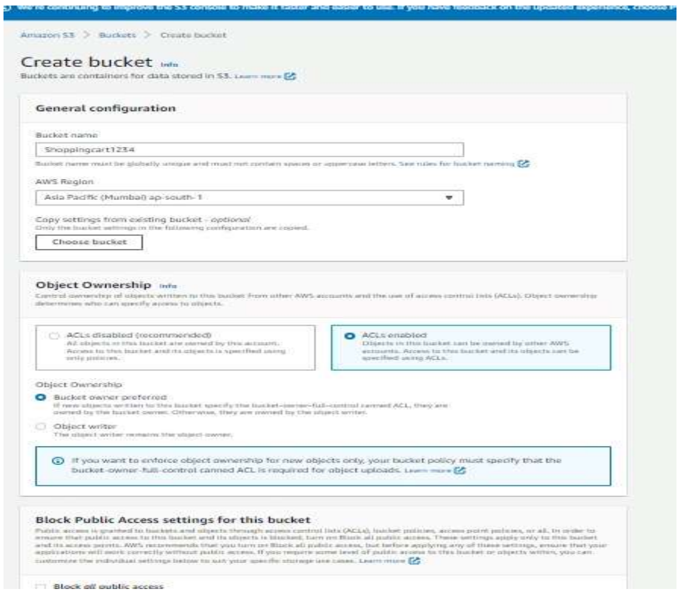
☑ Enable encryption

5°F
artly cloudy

Fig- Configurations of database

2.Creating S3 Bucket:

create a bucket in Amazon S3 using the AWS Management Console. Every object in Amazon S3 is stored in a *bucket*. Before you store data in Amazon S3, you must create a bucket.



Amazon S3 > Buckets > Create bucket

Create bucket Info

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name: Shoppingcart1234

Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming

AWS Region: Asia Pacific (Mumbai) ap-south-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Choose bucket

Object Ownership Info

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

- ACLs disabled (recommended) — All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.
- [✓ ACLs enabled] — Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Object Ownership

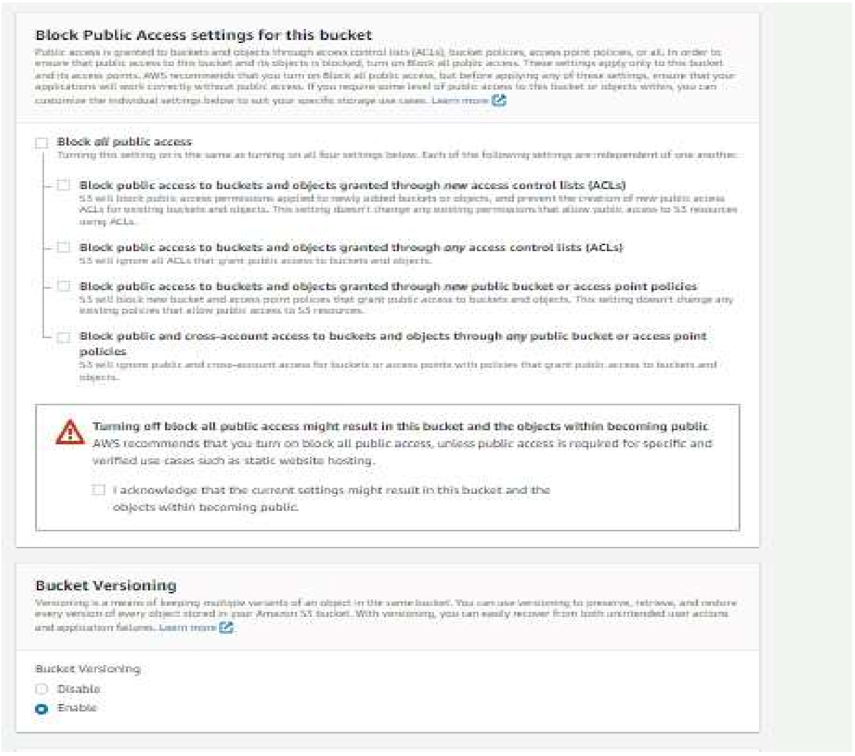
- [✓ Bucket owner preferred] — If new objects written to this bucket specify the bucket-owner-full-control canned ACL, they are owned by the bucket owner. Otherwise, they are owned by the object writer.
- Object writer — The object writer remains the object owner.

If you want to enforce object ownership for new objects only, your bucket policy must specify that the bucket-owner-full-control canned ACL is required for object uploads. Learn more

Block Public Access settings for this bucket

Block all public access

Fig- Creating Bucket



Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. Learn more

- ☐ Block *all* public access — Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.
  - ☐ Block public access to buckets and objects granted through *new* access control lists (ACLs) — S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.
  - ☐ Block public access to buckets and objects granted through *any* access control lists (ACLs) — S3 will ignore all ACLs that grant public access to buckets and objects.
  - ☐ Block public access to buckets and objects granted through *new* public bucket or access point policies — S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
  - ☐ Block public and cross-account access to buckets and objects through *any* public bucket or access point policies — S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.

Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

☐ I acknowledge that the current settings might result in this bucket and the objects within becoming public.

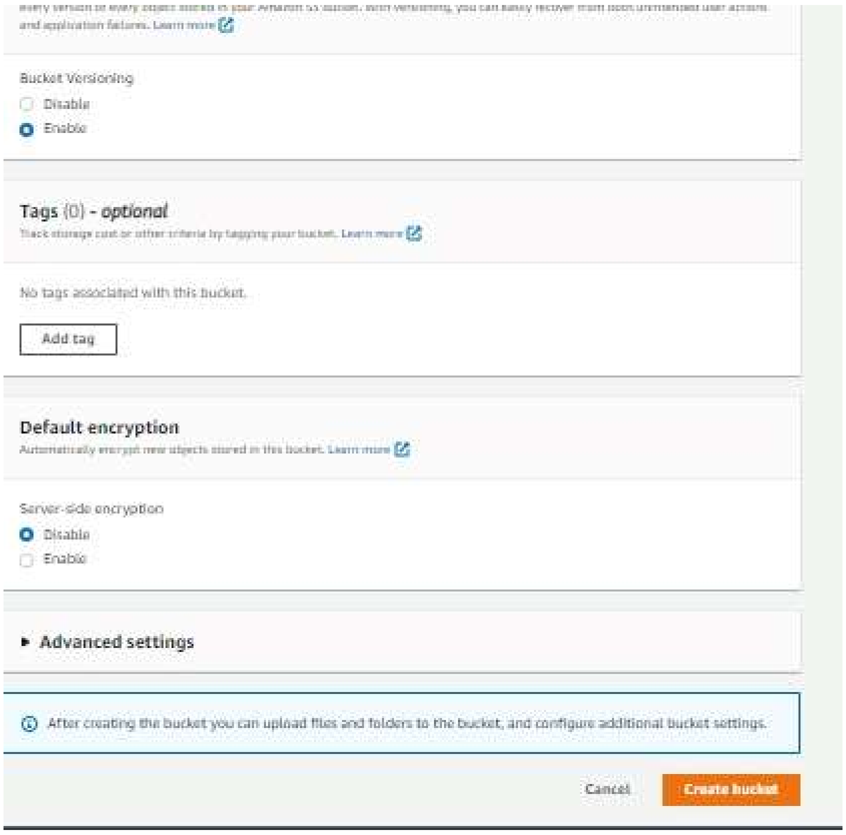
Bucket Versioning

Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

- Disable
- [✓ Enable]

Fig- Accessing of Bucket


every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more

Bucket Versioning

Disable [✓ Enable]

Tags (0) - *optional*

Track storage cost or other criteria by tagging your bucket. Learn more

No tags associated with this bucket.

Default encryption

Automatically encrypt new objects stored in this bucket. Learn more

Server-side encryption

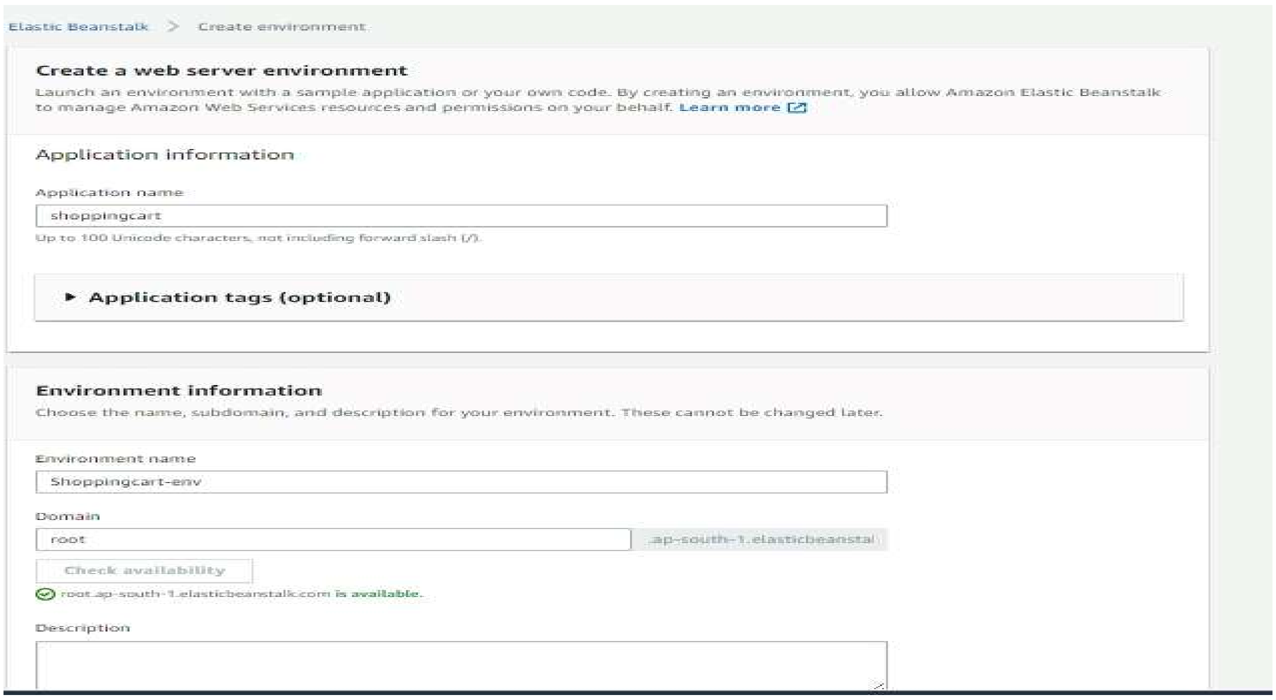
[✓ Disable] Enable

After creating the bucket you can upload files and folders to the bucket, and configure additional bucket settings.

Fig- Optional configurations of bucket

3. Creating elastic beanstalk environment:

An AWS Elastic Beanstalk *environment* is a collection of AWS resources running an application version. You can deploy multiple environments when you need to run multiple versions of an application. For example, you might have development, integration, and production environments.


Elastic Beanstalk > Create environment

Create a web server environment

Launch an environment with a sample application or your own code. By creating an environment, you allow Amazon Elastic Beanstalk to manage Amazon Web Services resources and permissions on your behalf. Learn more

Application information

Application name: shoppingcart

Up to 100 Unicode characters, not including forward slash (/).

▸ Application tags (optional)

Environment information

Choose the name, subdomain, and description for your environment. These cannot be changed later.

Environment name: Shoppingcart-env

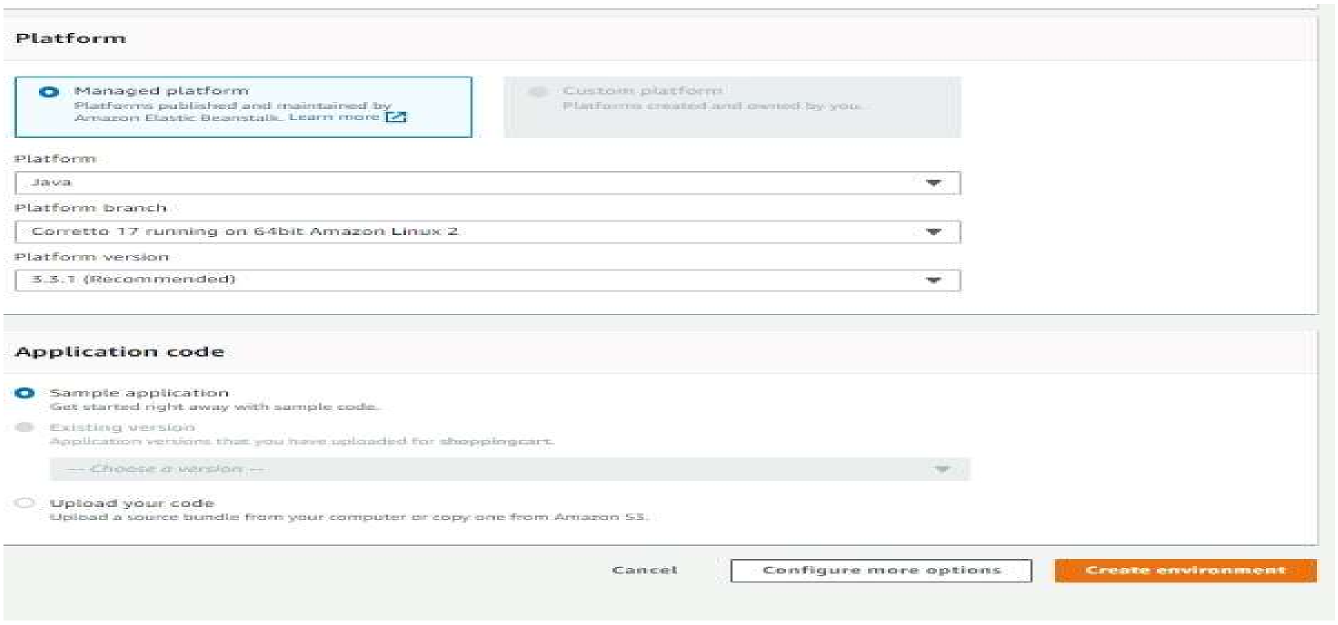
Domain: root .ap-south-1.elasticbeanstal

Check availability

root.ap-south-1.elasticbeanstalk.com is available.

Description

Fig- web server creation


Platform

● Managed platform — Platforms published and maintained by Amazon Elastic Beanstalk. Learn more

○ Custom platform — Platforms created and owned by you.

Platform: Java

Platform branch: Corretto 17 running on 64bit Amazon Linux 2

Platform version: 3.3.1 (Recommended)

Application code

● Sample application — Get started right away with sample code.

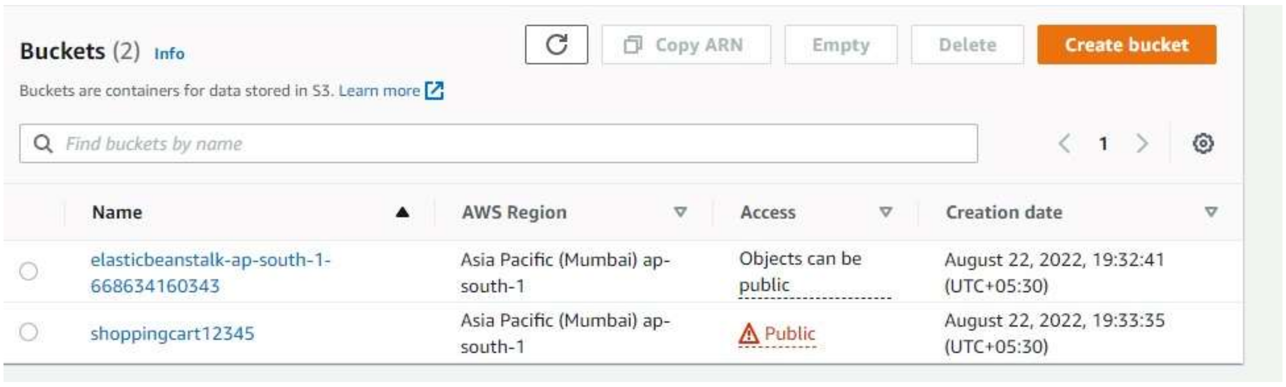
○ Existing version — Application versions that you have uploaded for shoppingcart.

-- Choose a version --

○ Upload your code — Upload a source bundle from your computer or copy one from Amazon S3.

Cancel | Configure more options | Create environment

Fig- creating an environment



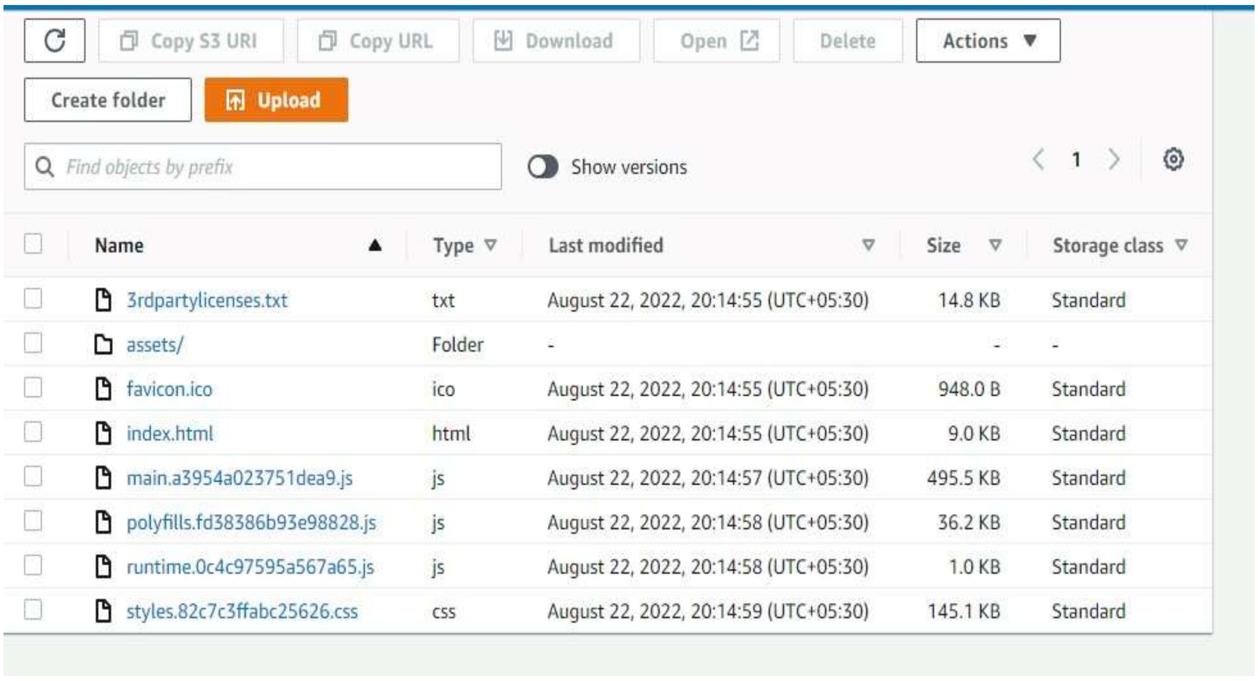
Buckets are containers for data stored in S3. Learn more

Name	AWS Region	Access	Creation date
elasticbeanstalk-ap-south-1-668634160343	Asia Pacific (Mumbai) ap-south-1	Objects can be public	August 22, 2022, 19:32:41 (UTC+05:30)
shoppingcart12345	Asia Pacific (Mumbai) ap-south-1	⚠ Public	August 22, 2022, 19:33:35 (UTC+05:30)

Fig- created elasticbeanstalk

3.Uploading data in s3 bucket:

When you upload a file to Amazon S3, it is stored as an S3 object. Objects consist of the file data and metadata that describes the object. You can have an unlimited number of objects in a bucket. You can upload any file type—images, backups, data, movies, etc.—into an S3 bucket. The maximum size of a file that you can upload by using the Amazon S3 console is 160 GB. To upload a file larger than 160 GB, use the AWS CLI, AWS SDK, or Amazon S3 REST API.

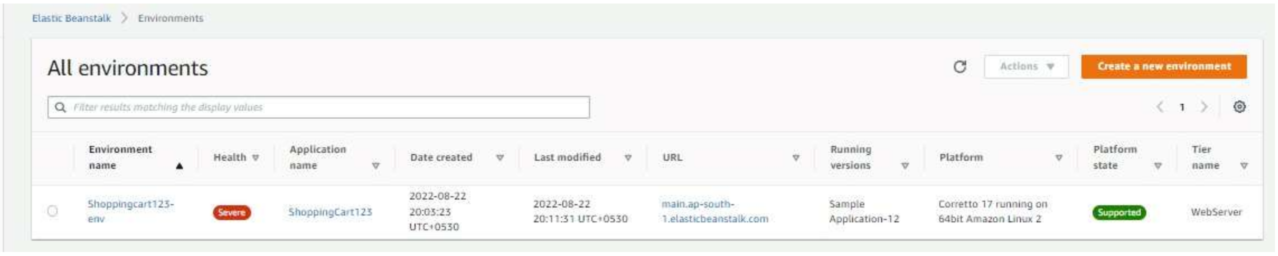


Name	Type	Last modified	Size	Storage class
3rdpartylicenses.txt	txt	August 22, 2022, 20:14:55 (UTC+05:30)	14.8 KB	Standard
assets/	Folder	-	-	-
favicon.ico	ico	August 22, 2022, 20:14:55 (UTC+05:30)	948.0 B	Standard
index.html	html	August 22, 2022, 20:14:55 (UTC+05:30)	9.0 KB	Standard
main.a3954a023751dea9.js	js	August 22, 2022, 20:14:57 (UTC+05:30)	495.5 KB	Standard
polyfills.fd38386b93e98828.js	js	August 22, 2022, 20:14:58 (UTC+05:30)	36.2 KB	Standard
runtime.0c4c97595a567a65.js	js	August 22, 2022, 20:14:58 (UTC+05:30)	1.0 KB	Standard
styles.82c7c3ffabc25626.css	css	August 22, 2022, 20:14:59 (UTC+05:30)	145.1 KB	Standard

Fig-Uploading data into bucket

5.Uploading data into Environment:

Here we upload data into environment as,



Elastic Beanstalk > Environments

All environments

Environment name	Health	Application name	Date created	Last modified	URL	Running versions	Platform	Platform state	Tier name
Shoppingcart123-env	Severe	ShoppingCart123	2022-08-22 20:03:23 UTC+0530	2022-08-22 20:11:31 UTC+0530	main.ap-south-1.elasticbeanstalk.com	Sample Application-12	Corretto 17 running on 64bit Amazon Linux 2	Supported	WebServer

Fig- Environment with data uploaded

OUTPUT

Final aws uploaded website :

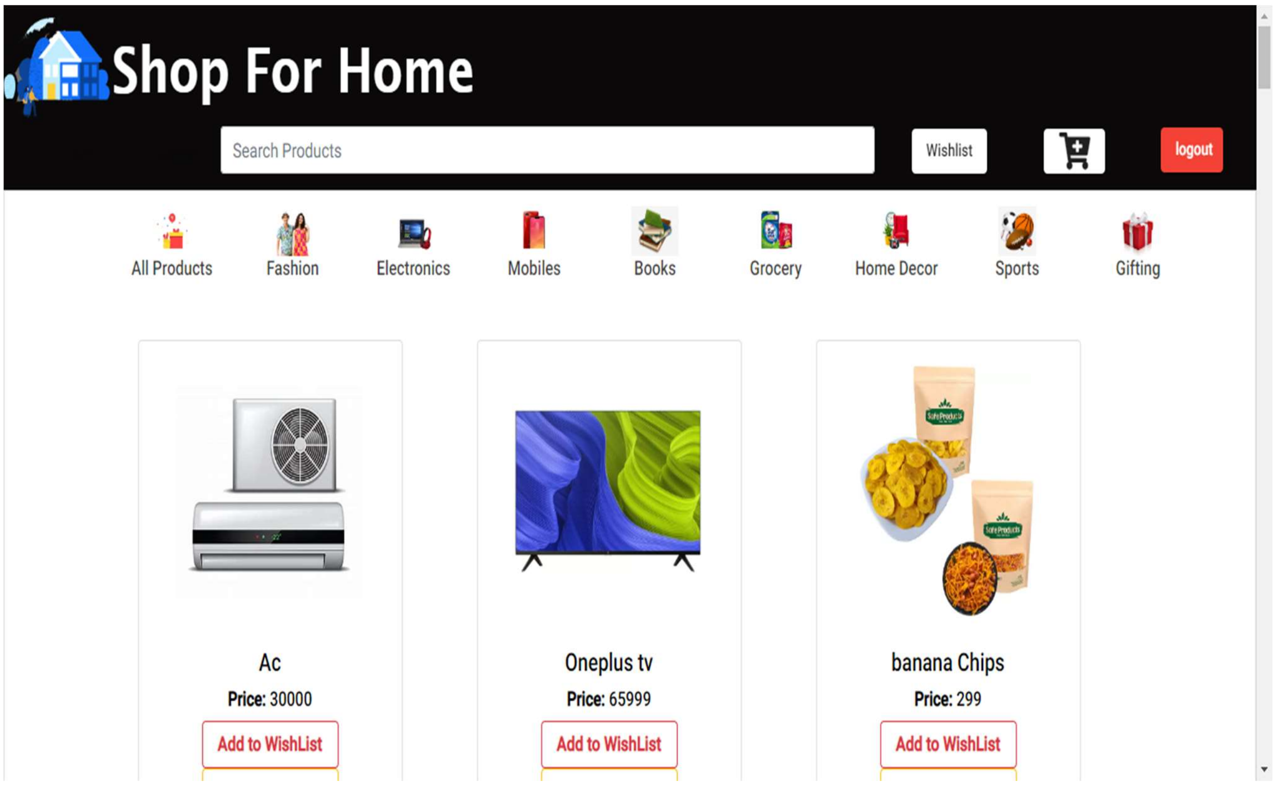


Fig-Home Page

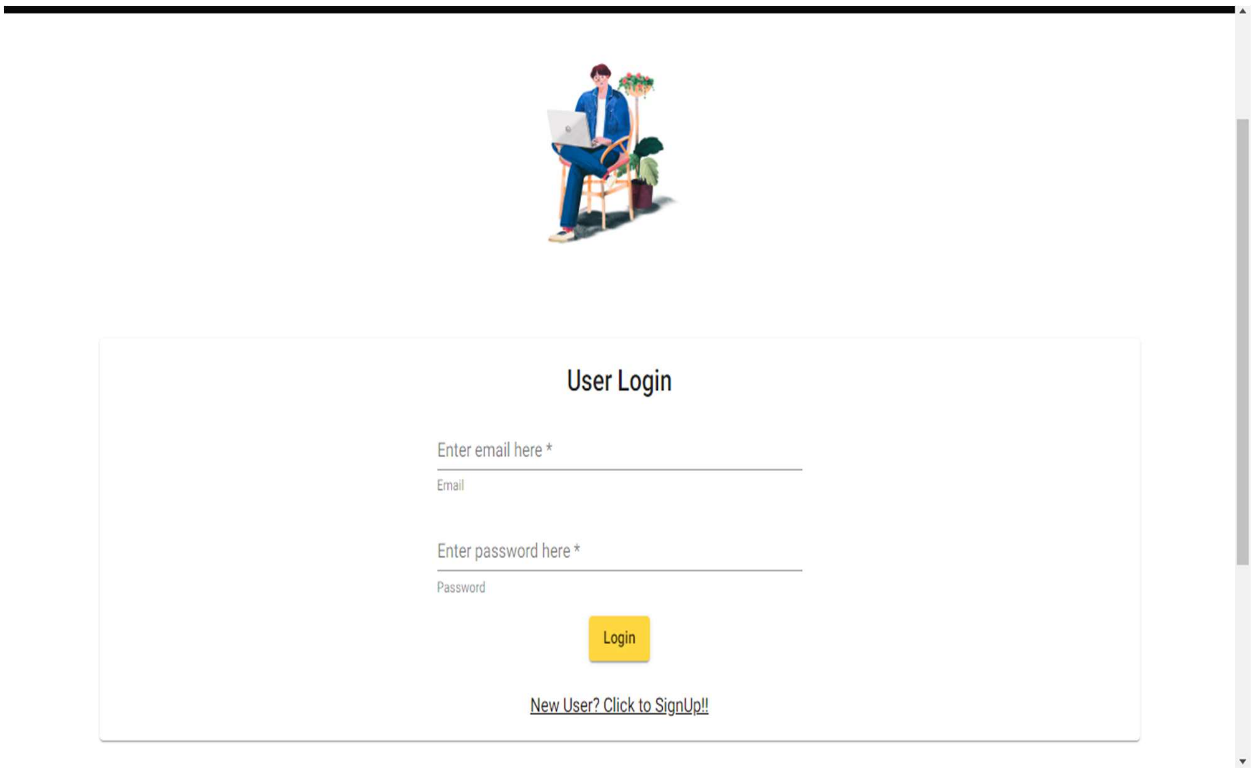


Fig- login page

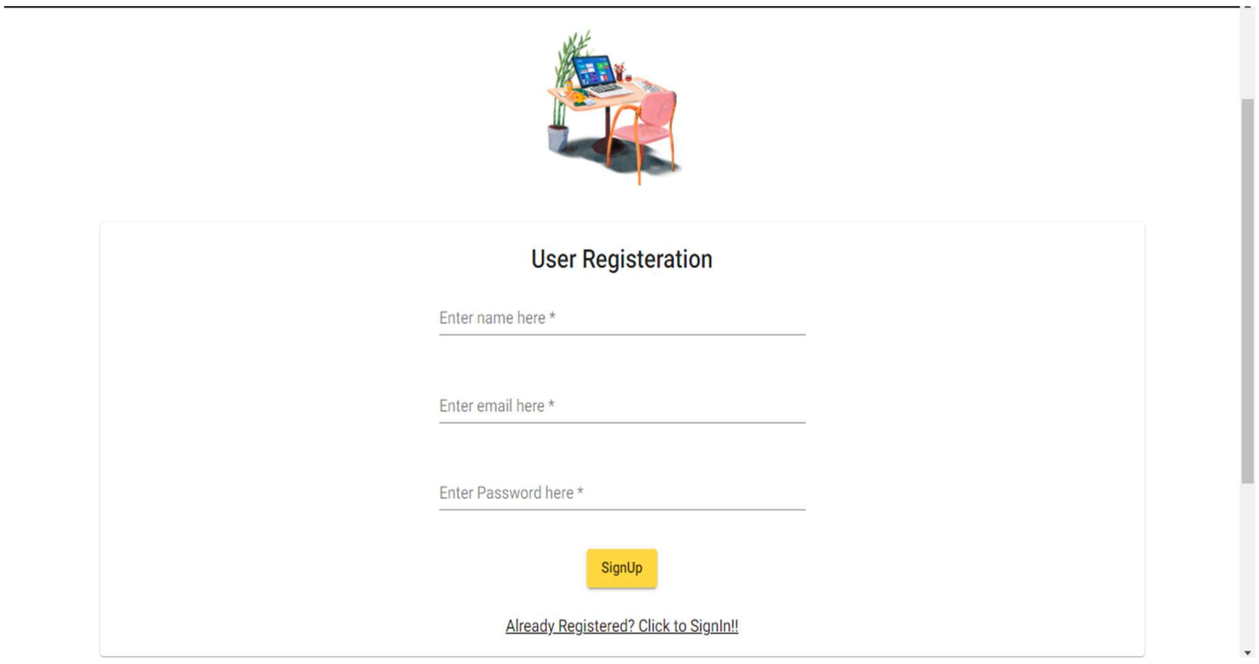


Fig-Registration Page

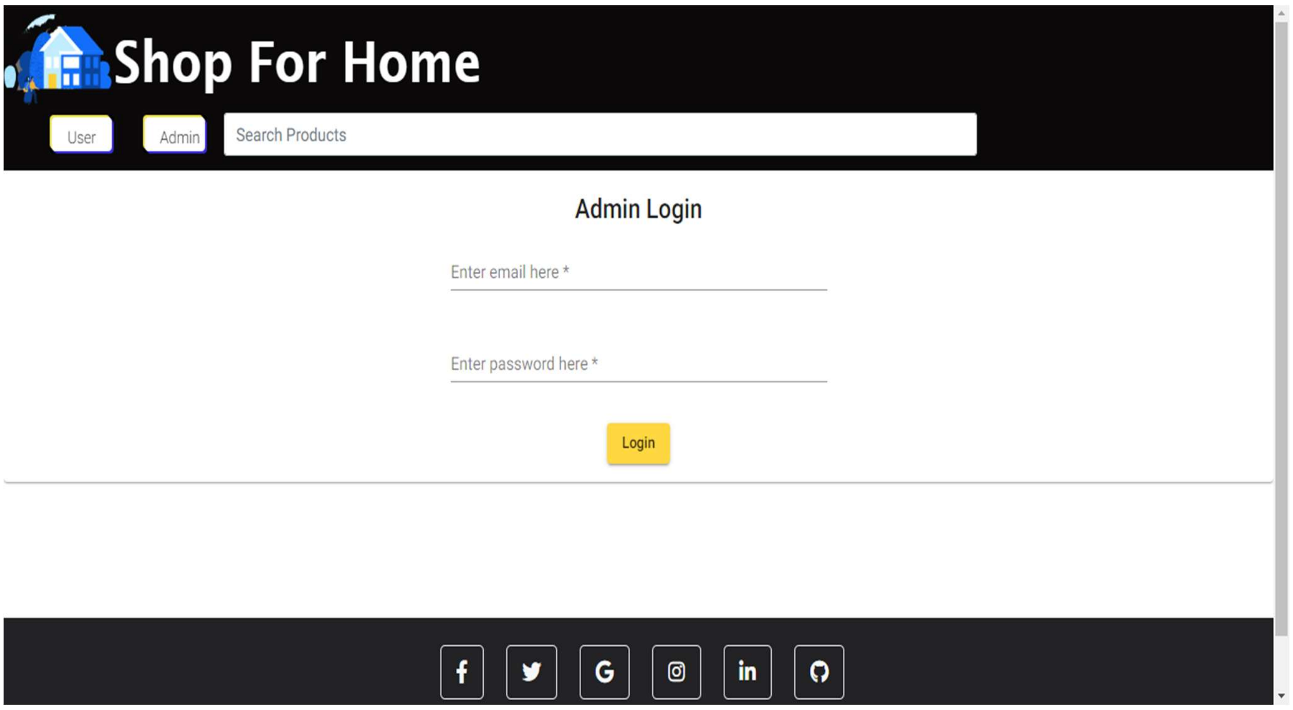
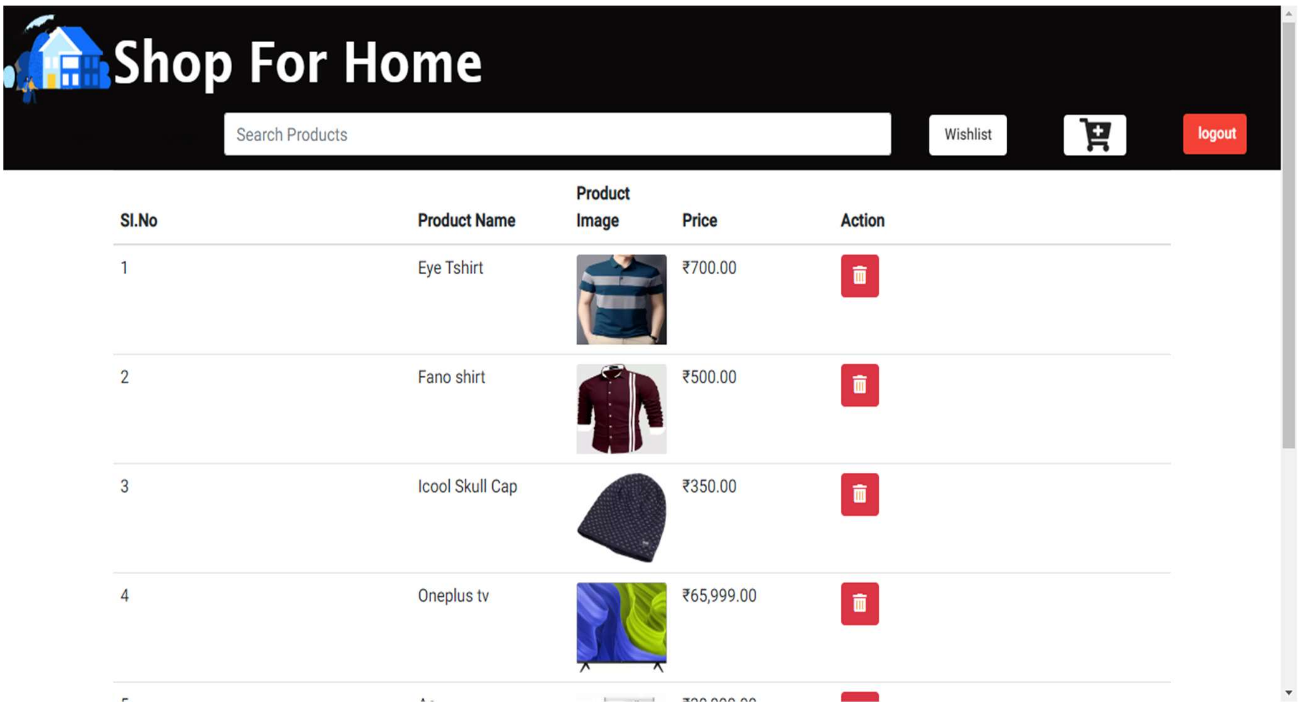
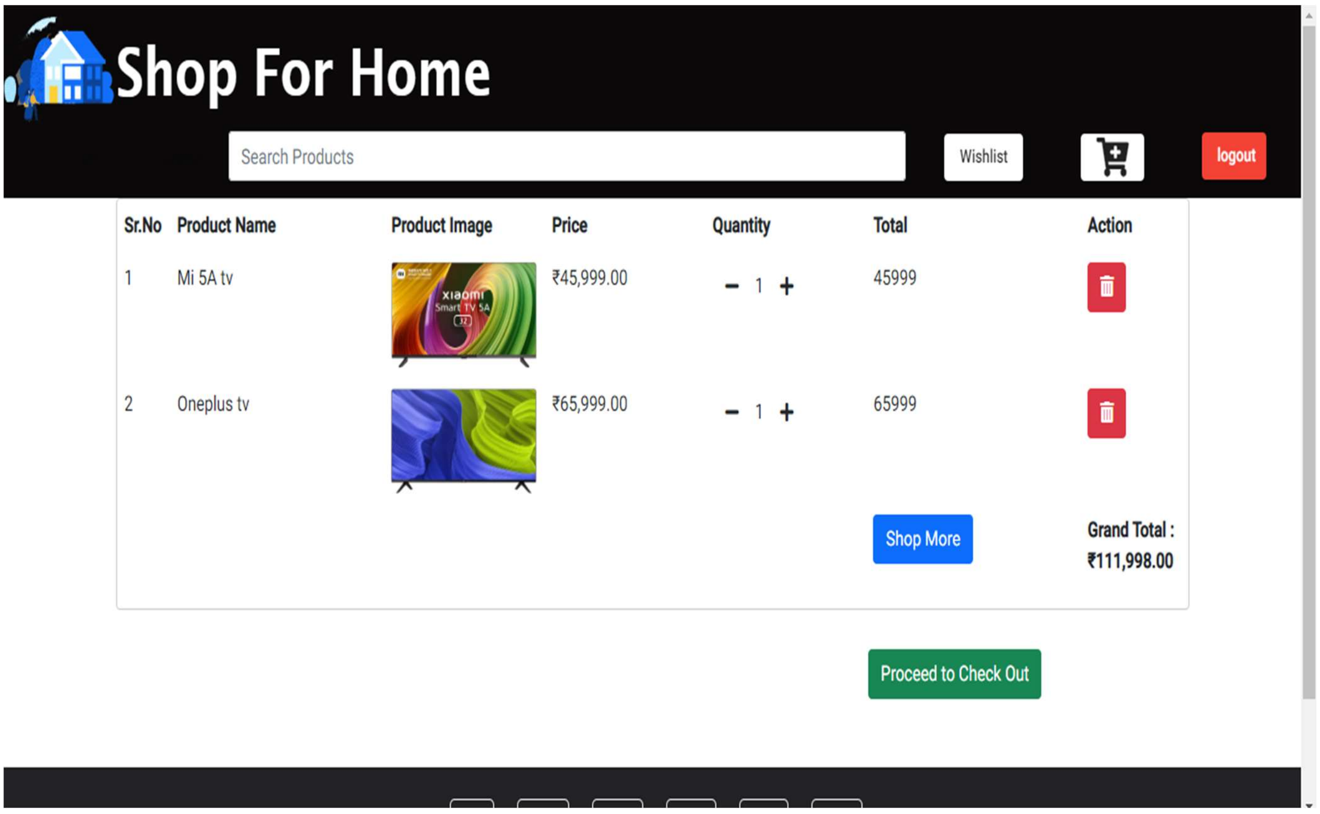


Fig- Admin Page



Sl.No	Product Name	Product Image	Price	Action
1	Eye Tshirt		₹700.00	
2	Fano shirt		₹500.00	
3	Icool Skull Cap		₹350.00	
4	Oneplus tv		₹65,999.00	

Fig-Wishlist Page



Sr.No	Product Name	Product Image	Price	Quantity	Total	Action
1	Mi 5A tv		₹45,999.00	− 1 +	45999	
2	Oneplus tv		₹65,999.00	− 1 +	65999	

Grand Total : ₹111,998.00

Fig-Cart page

Conclusion:

E-commerce is continuously progressing and is becoming more and more important to businesses as technology continues to advance and is something that should be taken advantage of and implemented. From the inception of the Internet and e-commerce, the possibilities have become endless for both businesses and consumers. Creating more opportunities for profit and advancements for businesses, while creating more options for consumers.

In general, today's businesses must always strive to create the next best thing that consumers will want because consumers continue to desire their products, services etc. to continuously be better, faster, and cheaper. In this world of new technology, businesses need to accommodate to the new types of consumer needs and trends because it will prove to be vital to their business' success and survival.

However, just like anything else, e-commerce has its disadvantages including consumer uncertainties, but nothing that cannot be resolved or avoided by good decision-making and business practices. There are several factors and variables that need to be considered and decided upon when starting an e-commerce business. Some of these include types of e-commerce, marketing strategies, and countless more. If the correct methods and practices are followed, a business will prosper in an e-commerce setting with much success and profitability.