

ShoppingNova

Group Name: ITCT

1st Sung-Mun Hong
dept. Computer Softwareline
HanYang Univ.
Seoul, Republic of Korea
sunmoonkr99@hanyang.ac.kr

2nd Ha-Eun Jung
dept. Information System
HanYang Univ.
Seoul, Republic of Korea
hyever15@hangyang.ac.kr

3rd Geon-U Kim
dept. Computer Softwareline
HanYang Univ.
Seoul, Republic of Korea
mambomul@hanyang.ac.kr

4th Seon-Woong Ha
dept. Information System
HanYang Univ.
Seoul, Republic of Korea
ha819ha@hanyang.ac.kr

Abstract—This paper explores a strategy to increase visitors to online shopping malls by introducing a visually distinctive concept. We propose the use of a galaxy-themed visual representation to differentiate the shopping experience. In this model, each product is represented as a star or planet, forming clusters (categories) of products, which then combine to create a galaxy-like shopping environment. The design will be implemented in a stepwise fashion, with each level revealing a new visual layout, enhancing the user's experience as they explore the shopping mall. We believe that this unique visual approach can attract more visitors by providing an engaging and memorable user interface.

Index Terms—Online shopping mall, visual distinction, galaxy concept, user interface design, visitor increase, shopping experience, product visualization, category clusters

Table I: Role Assignments

Role	Name	Task description and etc.
User	Geon-U Kim	Defining actual utility of the service and evaluating the value of the service by the user's point. Also giving continuous feedback which meets the user's requirements and helping the development of service.
Customer	Sung-Moon Hong	Giving opinions for developing a strategy to attract visitors by adding visual enjoyment to the shopping mall. Also criticizing captivation of the beauty of our online shopping mall, and determining whether to make a purchase.
Software Developer	Ha-Eun Jung	Developing software with analyzing requirements, implementing and testing software features, and maintaining and updating the software performance.
Development Manager	Seon-Woong Ha	Allocating the tasks to be done in the project to the schedule and selecting the development framework that fits each team member's development stack. Completing software by leading communication between team members. Managing the schedule using Notion and handles code maintenance through Git.

I. INTRODUCTION

A. Motivation

In modern online shopping malls, customer experience is prioritized beyond mere product sales and display. As internet users' attention spans continue to shorten, they seek fresh and engaging online experiences. Visually appealing design plays a critical role in shaping the customer's first impression, encouraging longer engagement on the site, and increasing the likelihood of purchase conversion. A user-friendly and aesthetically pleasing website not only differentiates a business in a competitive market but also enhances customer satisfaction and leaves a lasting impression.

In the highly competitive e-commerce landscape, a distinctive visual design is a powerful tool for fostering emotional connections with customers, surpassing the realm of simple aesthetic appeal. Moreover, a visually differentiated online shopping platform can positively impact the attraction of potential new customers. Therefore, it is both valuable and necessary to pursue improvements in the website's visual design to better engage and retain a broader customer base.

B. Problem Statement (Client's Need)

The current shopping mall website functions adequately in terms of providing product information and facilitating payment systems, but it lacks the visual design and user experience needed to attract a broader customer base. Specifically, there is little incentive for customers to visit the website unless they are explicitly intending to make a purchase. However, if the website could draw in potential customers through visual appeal, beyond those simply looking to buy, it would naturally increase product exposure and make a lasting impression. Additionally, aesthetically pleasing designs can enhance user engagement by encouraging visitors to spend more time on the site, thereby increasing the likelihood of conversion.

This indicates significant potential for growth in terms of customer acquisition and purchase conversion rates for the shopping mall website. Modern consumers seek emotional satisfaction through intuitive and sensory-driven experiences, and during this process, they tend to form long-term relationships with brands that leave a lasting impact. Therefore, the objective of this project is to enhance the website's

visual appeal, with the goal of attracting and retaining more customers over the long term.

C. Related Software

1) TheDropStore

TheDropStore is an online platform that sells a variety of kits to spread awareness about solving water crises. This platform uses 3D elements to provide a differentiated shopping experience. Each product is rendered in a 3D model, allowing the user to experience the real object as if they were looking at a real object. These features allow the customer to enjoy the product search process itself and naturally sympathize with the message about the water crisis.

2) IKEA Place

IKEA Place is an app that enables furniture to be placed in a virtual space using AR technology. Users can see how 3D furniture models are placed in the space by fusing them with their real environment. This allows users to intuitively identify the harmony of size, color, and design. However, this arrangement requires 3D model files, and if the size or ratio of the model is inaccurate, there is a drawback that can lead to a difference between the virtual and actual arrangements.

II. REQUIREMENT

A. User Interface & User Experience

Theme: The theme of the shopping mall webpage is inspired by space, incorporating elements that evoke vastness and mystery of the cosmos. The design features a vast and expansive background with accents of nebula-like colors and celestial imagery, creating an immersive experience that transports users to a universe beyond their own.

B. User Account Management

1) Login

Users can log in to the system using their user ID and password. If incorrect login details are provided, the system will prompt the user to retry entering their information.

2) Sign-up

Users can sign up by providing a unique user ID, password, name, phone number, and email. The system checks for duplicate user IDs to ensure each user has a unique ID, and passwords must meet specific length and character requirements.

3) Account Information Modification

Users can update their personal information, including their name, phone number, and email, after completing SMS or email verification. They have the ability to review

and modify their account information at any time. Additionally, users can view their purchase history, allowing them to check past purchases.

C. Product Management

1) Product Registration

Administrators can register new products by entering required details, such as product name, category, and brand. If the desired category or brand does not exist, administrators have the option to add new categories or brands.

2) Product Information Management

Administrators can update product details, including inventory levels and stock status. They can also delete products from the catalog and receive notifications when items are out of stock. Administrators have access to view detailed information for each product, such as the product's name, price, and current stock status.

3) Product Category

Users can visually browse through multiple categories on the screen. When a specific category is selected, a list of products associated with that category is displayed. By selecting a product, users can view its detailed information, including pricing, descriptions, and stock availability.

D. Shopping Cart

Users can add and edit their desired products and quantities in their shopping cart, allowing them to manage their selections efficiently. They have the capability to review their cart contents, select individual items for modification, and update quantities as needed. Additionally, users can remove items from their cart, ensuring they have control over their shopping experience.

E. Product Filtering

1) Search Functionality

Users can search for products that match their specified criteria.

2) Sorting Options

Users can sort the products in ascending or descending order based on their selected conditions.

F. Related Product Recommendation

While browsing, users can receive recommendations for other versions of products that may be of interest to them. This feature provides personalized suggestions based on browsing behavior and product compatibility, enriching the shopping experience by introducing users to alternatives they may not have considered.

G. Security

The policy for storing, accessing, and deleting user data must be clearly defined with careful consideration for privacy policies. This includes implementing secure data handling practices to protect user information, ensuring compliance with applicable regulations, and regularly reviewing security measures to adapt to evolving privacy standards.

H. Non-Functional

A UI that is not inconvenient for users to use should be configured. When expressed in 3D space, an algorithm for calculating the correlation between products is appropriately constructed so that related products may be well recommended.

III. DEVELOPMENT ENVIRONMENT

A. Choice of Software Development Platform

1) Programming Languages

- JAVA

Java is an open-source, object-oriented programming language developed by Sun Microsystems. Key features of Java include support for Object-Oriented Programming (OOP), automatic memory management through garbage collection, robust exception handling, and multi-threading capabilities. Java supports various frameworks and libraries, and we choose spring boot that support this language.

- Typescript

TypeScript is a strongly typed programming language that builds on JavaScript and was developed by Microsoft. As a superset of JavaScript, TypeScript is compatible with JavaScript code and includes the latest ECMAScript features. It is designed especially to improve code stability and maintainability at any scale. Key features of TypeScript include static type checking, interfaces, type inference, and support for classes and inheritance. These features make it suitable for stable development and maintain a high-level of code quality. We decided to use Next.js and have experience using typescript, so we chose this language.

2) Frameworks & Libraries

- Spring Boot

Spring Boot is a Java-based framework that allows rapid development of web applications without complex configurations. In the shopping mall project, Spring Boot serves as the backend, handling essential functions such as user authentication, product and order management, and shopping cart features. It communicates with the frontend via RESTful APIs, and integrates with the database through JPA (Java Persistence

API) and Hibernate for efficient CRUD functionality. Spring Boot's robust architecture ensures scalability and maintainability, making it ideal for large-scale applications.

- NextJS

Next.js is a React-based full-stack framework that supports both client-side and server-side rendering. In implementing the frontend of the shopping mall, Next.js enhances page load speed and SEO by pre-fetching data on the server side. Using this library, we build and deploy shopping mall page.

- Three.js & React-three-fiber

Three.js is a JavaScript-based 3D graphics library that enables rendering complex 3D scenes and animations in the browser. Operating on top of WebGL, Three.js provides a wide range of APIs for manipulating 3D objects, cameras, lighting, and textures without requiring developers to write direct WebGL code.

React-three-fiber(R3F) abstracts Three.js's capabilities into React components and hooks. With R3F, developers can utilize React's state management to easily control dynamic 3D objects, blending standard React components with 3D elements to create immersive user interfaces. Using this library, we will show stars and galaxy elements in a 3D viewer for the products and categories of the shopping mall.

3) Database

- MySQL

MySQL is a relational database management system (RDBMS) that is open-source, free, and suitable for both small and large applications. It is known for its speed, reliability, scalability, and ease of use. Unlike some RDBMS systems, MySQL does not come with built-in GUI management tools, so users typically rely on command-line interfaces or front-end desktop software and web applications to create, manage, back up, and interact with databases. The official MySQL front-end tool, MySQL Workbench, developed by Oracle, is freely available for use. MySQL is cross-platform and adheres to the ANSI SQL standard.

B. Software in Use

1) Visual Studio Code (Code Editor)

Visual studio code is a source code editor developed by Microsoft for Microsoft Windows, macOS, and Linux. It includes debugging support, Git control, and syntax emphasis functions. It is driven based on Electron framework developed by GitHub.

2) IntelliJ (Code Editor)

A code editor developed by JetBrains, widely used for Java development, with features like code completion, debugging, and refactoring. It also offers seamless integration with SQL databases, allowing developers to connect, query, and manage databases directly within the IDE.

3) Git/Github (Version Management)

A version control system (Git) and an online platform (GitHub) for managing and sharing code repositories, enabling collaboration and version tracking.

4) Docker (Containerization)

A platform for containerization that allows applications to run in isolated environments, making them easier to deploy and scale.

5) AWS EC2

Amazon Web Services' service for providing virtual servers in the cloud, enabling users to run applications on scalable infrastructure. For our shopping mall, we use EC2 to host the backend application, providing a reliable and scalable environment to handle user requests and manage traffic effectively.

6) AWS S3

Amazon Web Services' storage service for storing and distributing data (like images) securely in the cloud. We use S3 to store and manage product images and other media files for the shopping mall, ensuring fast and secure access to assets, with the ability to deliver them efficiently to users.

7) Figma

A design and prototyping tool that allows teams to collaborate in real-time on UI/UX designs, making it easy to create and share interactive mockups. Using Figma, we design the shopping mall pages, layout, and user flow. Additionally, real-time feedback from team members allows us to make design adjustments easily.

8) Notion

A versatile tool for task management, scheduling, and documentation, enabling teams to organize projects, track tasks, and store information in a collaborative workspace. With Notion, we can efficiently manage the project schedule, document requirements, and assign tasks. Team members can leave comments and update information in real-time.

9) Overleaf

A specialized online editor for writing documents in LaTeX, a typesetting system commonly used for academic and technical writing, offering precise control over

document formatting, especially for complex mathematical and scientific content.

10) Listly

A lightweight Google Chrome extension that brings an all-in-one experience to web data collection, trusted by more than 70,000 professionals worldwide. Also, it focuses on areas of unmet data cleaning needs and leverages its expertise to strive for solutions that streamline repetitive tasks and automate data collection processes. Shopping mall data for demo will be collected by Listly from LG Electronic shopping web site.

C. version

Name	System Environment
Geon-U Kim	macOS Sequoia 15.0 Node.js 20.18 , Next.js 15.0
Sung-Moon Hong	macOS Sequoia 15.0 Node.js 20.18 , Next.js 15.0
Ha-Eun Jung	Windows 11 Java SE 23, Spring boot 3.3, PostgreSQL 17
Seon-Woong Ha	Windows 10 Java SE 23, Spring boot 3.3, PostgreSQL 17

D. Task Distribution

Name	Task
Geon-U Kim	Front-End Development
Sung-Moon Hong	Front-End Development
Ha-Eun Jung	Back-End Development
Seon-Woong Ha	Back-End Development

IV. SPECIFICATION

A. Login Page

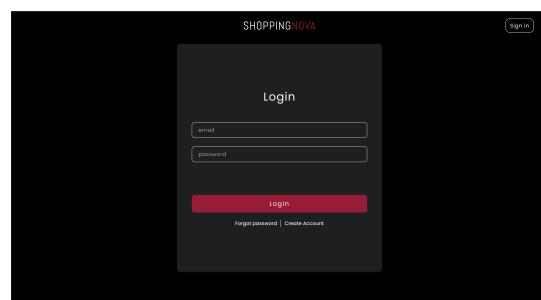


Figure 1: Login Page

1) Page Component

There are 2 input fields and 3 buttons in this page. Email input field is for the user to input their email address, and Password input field is for the user to input their account password. Login button initiates the login process by sending the entered email and password to the database for verification. Forgot Password Button

redirects the user to the password reset page. Create Account Button redirects the user to the sign-up page to create a new account.

2) UI Configuration

The Logo "SHOPPINGNOVA" is displayed at the top of the page. Below the logo, frame for login is shown that contains the Email and Password input fields, arranged vertically. Forgot Password and Create Account Buttons are positioned horizontally and below the input fields inside the frame.

3) Functionality

When User enters their Email and Password into the respective fields and clicks the login button, the input values are sent as a query to the database to retrieve user information. The database checks if the provided Email and Password match the stored values. If the credentials match, the user is redirected to the main page. If they don't match, a popup displays an error message indicating login failure.

4) Input Value format

Email should follow a valid email format and not exceed 255 characters in length. Password must be between 5 and 255 characters in length.

B. Sign-Up Page

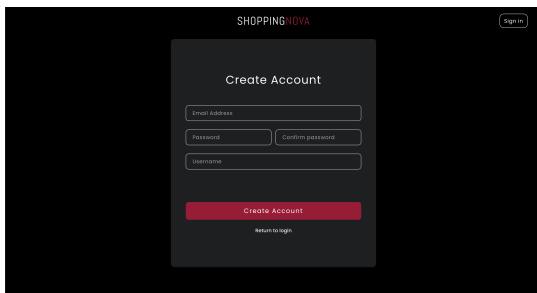


Figure 2: Sign-Up Page

1) Page Component

There are 4 input fields and 2 buttons on this page. The Email input field allows users to enter their email address, while the Password input field allows users to enter their account password. The Confirm Password input field is used to verify that the entered password matches. The Username input field allows users to enter their username. There is a Create Account Button that initiates the account creation process by sending the entered details to the database for validation, and a Return to Login Button that redirects the user back to the login page.

2) UI Configuration

The logo "SHOPPINGNOVA" is displayed at the top of the page. Below the logo, a frame is shown containing the input fields arranged vertically: Email, Password, Confirm Password, and Username. The Create Account Button and Return to Login Button are positioned horizontally below the input fields within the frame.

3) Functionality

When the user enters their Email, Password, Confirm Password, and Username and clicks the Create Account Button, the input values are sent to the database for validation. If all fields are properly formatted and meet the requirements, the account is successfully created. If there are any errors, a popup displays error messages indicating what needs to be corrected. The page also checks for duplicate user IDs, encrypts the password securely, and validates the format of email, phone numbers, and date of birth. If all input is valid, the database generates a unique user number and securely stores the password. Optionally, a welcome email or SMS notification may be sent to confirm successful registration. If the user clicks the Return to Login Button, they are redirected to the login page.

4) Input Value format

The Email must follow a valid email format and not exceed 255 characters in length. The Password must be between 5 and 255 characters in length, and the Confirm Password must match the entered Password exactly. The Username must be between 1 to 255 characters if entered in English, or 1 to 85 characters if entered in Korean.

C. Main Page (Category Page)

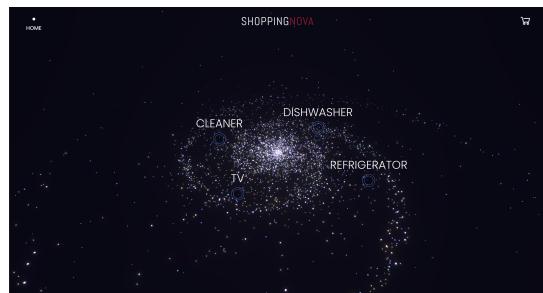


Figure 3: Category Page

1) Page Component

After logging in, four category buttons, a Cart button, and a Home indicator are displayed. Before logging in, the Cart button is replaced by a "Sign In" button.

2) UI Configuration

Below the logo "SHOPPINGNOVA," a glowing, revolving galaxy is displayed. The categories and buttons are fixed to this galaxy and rotate along with it. At the

top-left, the "HOME" label is visible, indicating that no higher-level page exists. On the top-right, the Cart button is present, indicating the user's cart.

3) Functionality

When the user clicks on any of the rotating category buttons below the revolving galaxy, they are smoothly navigated into the corresponding page of subcategories. For example, selecting the "Galaxy" category takes the user to the "Galaxy" subcategory, as the back-end retrieves and displays the corresponding Layer 2 (child) categories. Selecting a Layer 2 subcategory reveals the final Layer 3 - product cluster. The Cart button redirects logged-in users to their cart page. If the user is not logged in, the button is replaced by the "Sign In" button.

D. Product Cluster Page



Figure 4: Product Cluster Page

1) Page Component

This page includes a Page Layer indicator, a Cart Button, and a product information popup. The Page Layer indicator helps users track their navigation path, while the Cart Button allows quick access to the cart. Additionally, users can interact with product stars, which represent individual products displayed within the galaxy interface.

2) UI Configuration

At the top-left corner of the page, the Page Layer indicator displays the current navigation path, such as HOME → TV → ULTRA HD, helping users understand the hierarchy of categories they have navigated through. The Cart Button is positioned at the top-right corner, providing a direct link to the cart page. Below the logo, a galaxy with rotating product stars is displayed. Users can also interact with product stars to view a popup containing basic product information, including the product name, price, rating, and number of reviews.

3) Functionality

The Page Layer indicator allows users to track their navigation path and return to any previous layer by clicking the corresponding category. For instance, clicking TV will redirect the user to the TV category page, while

clicking HOME will navigate back to the main page. The Cart Button redirects the user to the cart page. Users can rotate the galaxy and product stars in 360 degrees and zoom in or out for a better view. When hovering over a product star, a popup appears showing basic product details such as name, price, rating, and number of reviews. Clicking on a product star focuses the view on the selected product and navigates to its detailed product page.

E. Product Detail Page

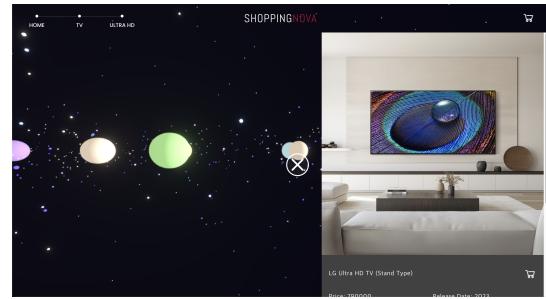


Figure 5: Product Detail Page 1

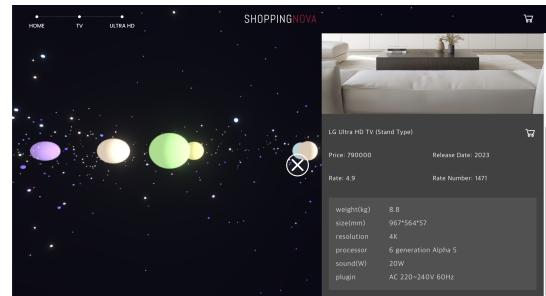


Figure 6: Product Detail Page 2

1) Page Component

The page contains a Page Layer indicator, a Cart Button, a product detail information frame, and a Close Button. The Page Layer indicator helps users track their navigation path, while the Cart Button allows easy access to the user's shopping cart. The product detail frame displays detailed information about the selected product, and the Close Button enables users to exit the detail view.

2) UI Configuration

The Page Layer indicator is positioned in the top-left corner, and the Cart Button is located in the top-right corner. Below the logo, the selected product star is prominently displayed at the center of a galaxy-themed background. On the right side, there is a scrollable product detail frame that displays various product specifications, such as name, price, release year, rating, number of reviews, weight, and category-specific details

like resolution for TVs or energy efficiency ratings for refrigerators. The frame includes a Cart Button for adding the product to the user's shopping cart and a Close Button on the left side to close the detail view.

3) Functionality

This page is displayed when a user clicks on a product star. The scrollable product detail frame allows users to explore all information retrieved from the database about the selected product. Different categories display unique specifications relevant to their type. For example, TVs show details like resolution and AI processor, while refrigerators include color and energy efficiency ratings. Clicking the Cart Button within the frame adds the product to the user's shopping cart. Clicking the Close Button exits the product detail view and navigates the user back to the product cluster page.

F. Shopping Cart Page

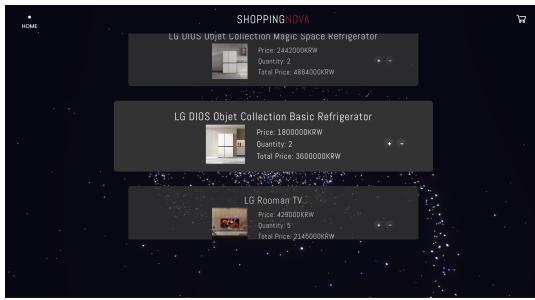


Figure 7: Shopping Cart Page

1) Page Component

The page consists of a HOME button, a Cart Button, a scrollable frame displaying cart items, and + (add) and - (delete) buttons within each item's frame.

2) UI Configuration

The HOME button is positioned at the top-left corner of the page. Cart items are displayed in a vertically scrollable layout against a galaxy-themed background. The cart item centered in the frame is emphasized with a larger size for better visibility. Each item frame includes the product name, image, price, quantity, and total price, which are centrally aligned. The + (add) and - (delete) buttons are displayed on the right side of each item frame.

3) Functionality

The HOME button in the top-left corner redirects users to the main page, exiting the cart page. Users can view all products in their cart along with the total amount. Cart items can be scrolled vertically, with the focused item appearing larger for readability. Clicking the + (add) or - (delete) buttons adjusts the quantity of the respective item by interacting with the back-end, which accesses the S3

database to update the cart information. The maximum quantity for any product is limited to 5. If the item quantity reaches 1 and the user clicks the - (delete) button, the item is removed from the cart. The system also recognizes existing items in the cart and prevents duplicate additions.

V. ARCHITECTURE DESIGN & IMPLEMENTATION

A. Overall Architecture

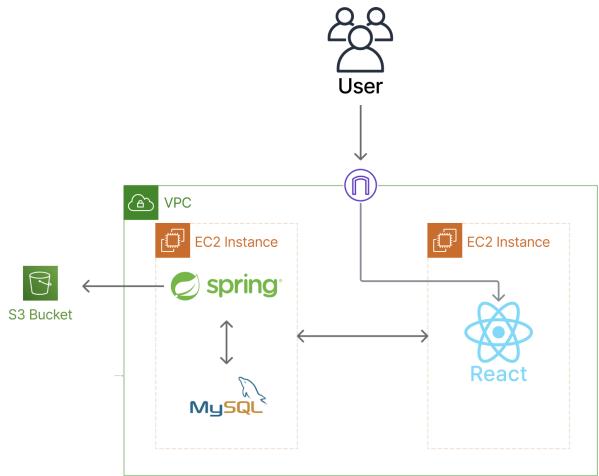


Figure 8: System Architecture

This system architecture demonstrates a web application hosted on AWS infrastructure with a React frontend, Spring Boot backend, and MySQL database. Static files, including product images and JSON data, are stored in AWS S3. The system operates within a secure VPC, utilizing EC2 instances for frontend and backend services.

Frontend (React) deployed on an EC2 instance, handles user interactions, and communicates with the backend via APIs.

Backend (Spring Boot) deployed on another EC2 instance, processes business logic, retrieves data from MySQL and S3, and delivers it to the frontend in JSON format.

Database (MySQL) stores structured data such as products, categories, users, and cart information. AWS S3 stores static files like product images and raw JSON data. The backend fetches, processes, and transforms these files.

VPC ensures secure communication between all components.

B. Database Structure

1) Entity Relationship Diagram

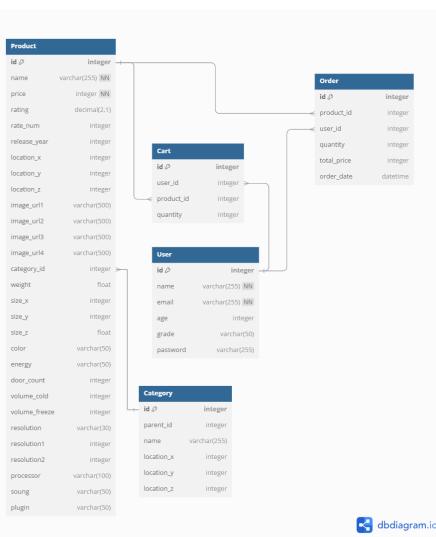


Figure 9: Entity Relationship Diagram

This ER diagram illustrates the database schema for a shopping system, detailing the relationships and roles of the main tables: Product, Category, User, Cart, and Order. Product table stores product details, including category, images, and specifications. Category table defines hierarchical product categories. User table stores user information like name, email, and grade. Cart table links users to products added to their shopping cart, including quantity. Order table tracks completed purchases, including product, user, total price, and date. Products belong to categories by category_id. Carts connect users and products. Orders track purchased products by users.

2) File Structure in AWS S3

```

S3 bucket "shopping-nova-1"
├── folder "image"
│   ├── product_tv_1_1.png
│   ├── product_tv_1_2.png
│   └── ...
├── folder "product_data"
│   ├── product_tv_1.json
│   ├── product_tv_2.json
│   └── ...
├── folder "user"
│   ├── user_1.json
│   ├── user_1_cart.json
│   └── ...
└── category.json

```

3) Data Collection & Processing

- Web Scraping by Listly (Chrome Extension)

For the LG Electronics shopping mall's new UI application project, product information, particularly for TVs and refrigerators, was scraped from the LG Electronics shopping mall webpage using Listly (a Chrome Extension) and saved as an Excel file.

- Data Cleaning & Wrangling by Excel (Manually)

The scraped data was manually cleaned and wrangled using Excel. Rows with null values in essential fields were removed, and only the necessary columns were retained.

- Converting & Formatting by Python (File I/O)

The Excel data was converted into JSON format using Python. Additionally, image URLs in AVIF format were processed and converted to PNG format with properly formatted file names.

- Uploading folders (json & image) to S3 bucket

Finally, the image, product_data, and users folders, along with the category.json file, were uploaded to an S3 bucket.

C. Directory Organization

1) Front-End

Directory	File Names	Modules in Use
shoppingnova-front/	.gitignore appspec.yml deploy.sh eslint.config.js index.html package.json package-lock.json postcss.config.js README.md tailwind.config.ts tsconfig.json tsconfig.app.json tsconfig.node.json vite.config.ts	vite postcss tailwindcss eslint
shoppingnova-front/.github/workflows	main.yml	
shoppingnova-front/src	index.css main.tsx vite-env.d.ts	react react-dom react-router-dom
shoppingnova-front/src/assets	particle.png	

shoppingnova-front/src/components	main cart detail products	react styled-components		
shoppingnova-front/src/components/three	Galaxy.tsx GalaxyLayer.tsx OrbitControls.tsx ThreeCanvas.tsx	react @react-three/drei @react-three/fiber		
shoppingnova-front/src/glsl	bpiral.gsls sprial.gsls frament.gsls gsls.d.ts			
shoppingnova-front/src/routes	Style index.tsx login.tsx register.tsx router.tsx	react-router-dom styled-components		
shoppingnova-front/src/utils/three	color.tsx rand.ts			

2) Back-End

Directory	File Names	Modules in Use
ShoppingNova-BE/src/java/Controller	CartController.java CategoryController.java ProductController.java UserController.java	spring-web jackson-databind aws-java-sdk-s3 java.util jackson-core
ShoppingNova-BE/src/java/Entity/Cart	Cart.java CartRepository.java CartService.java	spring-data-jpa jakarta-persistence spring-context spring-tx java.util java.time
ShoppingNova-BE/src/java/Entity/Category	Category.java CategoryRepository.java CategoryService.java	spring-data-jpa spring-web jackson-databind lombok
ShoppingNova-BE/src/java/Entity/Product	Product.java ProductRepository.java ProductService.java	spring-data-jpa spring-web jackson-databind lombok spring-tx
ShoppingNova-BE/src/java/Entity/User	User.java UserRepository.java UserService.java	spring-web jackson-databind aws-java-sdk-s3 java.util jackson-core

ShoppingNova-BE/src/java/Initializer	DataInitializer.java WebConfig.java	aws-java-sdk-s3 spring-web spring-context spring-data-jpa jackson-databind jackson-core java-util
ShoppingNova-BE/src/java/S3	S3Service.java	aws-java-sdk-s3 java-util-stream java-io aws-sdk-core

D. Front-End

1) purpose

We aim to create an unforgettable shopping experience using Three.js. Unlike typical 2D shopping websites, ShoppingNova offers users the ability to interact with products displayed as 3D objects in the browser. The frontend is developed using React and React-Three-Fiber(R3F), a library that integrates Three.js into React. R3F allows React developers to render Three.js components without disrupting the React ecosystem. This application visualizes product data from the server in 3D, presenting it as stars in a space-like universe for users to explore.

2) functionality

Displaying categories on a galactic space view, Navigate to another galaxy-like space by moving the camera when entering a category, Render product information as stars within a subcategory, Showing detailed product information when a star is clicked, Providing functions about adding Products to shopping cart, Managing the cart. Additional features such as user sign-up and login.

3) location of source code

github.com/hyu-cse4006/shoppingnova-front/tree/main

4) class components

- ShoppingNova-front/: This is the main directory for frontend source code of Shopping-Nova Service.
appspec.yml: The file defines deployment task and script execution steps for AWS CodeDeploy. Using CodeDeploy, we can automate post-deployment process.
eslint.config.js: The file specifies code style rules and linting for ESLint.
tsconfig.json: The file configures typescript compilation behavior.
vite.config.ts: The file defines plugins and build options of Vite. Vite optimizes build outputs.
- ShoppingNova-front/.github/workflows:

main.yml: The file defines CI/CD workflows in Github Actions. We use Github Actions for automating building, deploying the code when main branch is pushed.

- ShoppingNova-front/assets: This directory contains static assets like images and other media files that might be used in this app

- ShoppingNova-front/src: This is the main directory for react & typescript source code.

main.tsx: This is the entry point for React application. Use react-router-dom/RouteProvider to render proper page components.

index.css: The file contains global CSS styles and integrates Tailwind CSS.

vite-env.d.ts: The file is Typescript declaration file for extending Vite's module types.

- ShoppingNova-front/src/components: Contains reusable React components grouped by functionality.

- ShoppingNova-front/src/components/main: Components for the main view of the application : e.g., homepage or dashboard.

- ShoppingNova-front/src/components/cart: Components related to the shopping cart view, managing cart items and checkout features.

- ShoppingNova-front/src/components/detail: Components for the product detail view, providing a detailed display of individual products.

- ShoppingNova-front/src/components/products: Contains components for displaying product lists, grids, or galleries.

- ShoppingNova-front/src/components/three: Components related to 3D rendering and galaxy visualization using three.js.

- ShoppingNova-front/src/gsls: Contains WebGL shader files used by three.js for rendering custom graphics or animations.

fragment.gsls: The file defines pixel-level effects in WebGL rendering. Using assets/particle.png, we define the opacity of pixels.

bspiral.gsls: The file creates spiral galaxy graphics with tens of thousands of particles.

- ShoppingNova-front/src/routes: Page components corresponding to different routes in the application.

- ShoppingNova-front/src/utils: Utility functions, constants, and helper modules to simplify repetitive tasks or manage configurations.

- ShoppingNova-front/src/utils/three:

rand.ts: The file provides functions to randomize the star position in galaxy.

color.ts: The file contains functions for assigning colors based on star temperatures and other math utilities.

E. Back-End

1) Controller

- purpose

Provides APIs to deliver cart, category, product, user data stored in the database to the frontend as JSON files, formatted according to the requested type.

- functionality

Cart: Allows users to add products to their shopping cart. Retrieves the list of items in a user's cart, including product details and quantities. Enables removal of specific items in the cart.

Category: Returns a list of all product categories as JSON. Retrieves all products associated with a specific category.

Product: Returns all products belonging to a specific category as a single JSON file. Delivers detailed information of a specific product as a JSON file. Accepts a list of product IDs from the user's cart and returns either full or summary data for those products as JSON. Sorts and filters products based on price and ratings, and delivers the results as JSON.

User: Provides login functionality. Handles user sign-up by saving user details securely.

- location of source code

ShoppingNova-BE/src/java/Controller

- class components

Defines API endpoints.

Calls service layers to process data before returning JSON responses.

- where it's taken from

Fetches data from the database and processes it via ProductService.

- how/why you use it

Serves as the API to provide cart, category, product, user data in JSON format for frontend applications.

2) Cart Entity

- purpose

CartService: Manages shopping cart functionalities such as adding and removing items.

Cart: Defines an entity class mapping the shopping cart data.

CartRepository: Provides database access for cart-related operations using JPA's built-in functionalities.

- functionality

CartService: Implements core cart functionalities, such as fetching cart data and removing the cart.

Cart: Maps cart-related data, including user ID, product ID, and item quantity, to the database.

CartRepository: Handles CRUD operations on cart-related data.

- location of source code
ShoppingNova-BE/src/java/Entity/Cart
- class components
Logic for managing cart items.
Provides connections to other services, such as UserService and ProductService, to fetch related data.
- where it's taken from
Data originates from the user's session or database interactions.
- how/why you use it
Used to manage shopping cart functionality, allowing users to interact with and maintain their carts seamlessly.

3) Category Entity

- purpose
CategoryService: Handles logic for retrieving and organizing product categories into hierarchical structures for frontend use.
Category: Defines an entity class representing product categories, including parent-child relationships for hierarchical data.
CategoryRepository: Provides database access to fetch and manage category data using JPA.
- functionality
CategoryService: Retrieves all categories, structured into parent and child categories. Fetches child categories for a specific parent category. Provides APIs to retrieve categories along with associated product counts or summaries.
Category: Maps category data, including category ID, name, and parent-child relationships, to the database.
CategoryRepository: Performs CRUD operations and custom queries for hierarchical category database.
- location of source code
ShoppingNova-BE/src/java/Entity/Category
- class components
Handles relationships between parent and child categories.
Provides methods for returning structured category data for frontend display.
Fetches product data grouped or filtered by categories.
- where it's taken from
Data is stored in a relational database, with parent-child relationships established via parent_id.
- how/why you use it
Provides hierarchical category data to display menus or category lists on the frontend. Enables users to navigate and filter products by parent or child categories. Simplifies browsing by organizing products into logical groupings.

4) Product Entity

- purpose
ProductService: Manages the transfer of product information in JSON format from S3 to MySQL DB and provides essential JPA-based services.
Product: Defines an entity class mapping the product information retrieved from S3.
ProductRepository: Provides database access using JPA's built-in functionalities.
- functionality
ProductService: Retrieves product data from S3 and stores it in the database. Implements basic CRUD operations and data processing logic using JPA.
Product: Maps data to the Product entity.
ProductRepository: Handles CRUD operations on the database.
- location of source code
ShoppingNova-BE/src/java/Entity/Product
- class components
Logic for saving data to the database.
Invokes S3Service to fetch data from S3.
- where it's taken from
Data originates from JSON files stored in S3 and is saved to MySQL DB.
- how/why you use it
Facilitates the storage and processing of product data for further use in APIs or business logic.

5) User Entity

- purpose
UserService: Manages user-related data and operations, including registration, authentication, and profile management.
User: Defines an entity class mapping user data stored in the database.
UserRepository: Provides database access for user-related operations using JPA.
- functionality
UserService: Handles user registration, login, and other user-specific features.
User: Maps user data, such as username, password, and others to the database.
UserRepository: Handles CRUD operations for user-related database.
- location of source code
ShoppingNova-BE/src/java/Entity/User
- class components
Logic for user registration, login, and profile management.
- where it's taken from

Data originates from user input or database interactions.

- how/why you use it

Used to manage user data and authentication, ensuring a personalized and secure shopping experience.

6) S3

- purpose

Provides functionalities for interacting with AWS S3, such as uploading, downloading, extracting file lists, and generating URLs.

- functionality

File List Extraction: Used by ProductService to retrieve all product information.

URL Conversion: Converts image data into URLs (planned for API extension).

Upload/Download: Not implemented in the current project.

- location of source code

ShoppingNova-BE/src/java/S3

- class components

Methods for interacting with AWS S3.

Utilizes AWS SDK for S3 operations.

- where it's taken from

Pulls data from the AWS S3 bucket.

- how/why you use it

Supports product data management and generates image URLs for frontend consumption.

F. Database

1) purpose

Serve as a centralized storage system that integrates data from AWS S3 and organizes it into relational tables for efficient querying and API responses. Transform JSON and PNG files stored in S3 into structured data for backend processing and frontend consumption.

2) functionality

Data Extraction: Retrieve JSON data and image files from AWS S3. Extract and parse the JSON data into database tables (e.g., Product, Cart, Category, User).

Data Transformation and Storage: Map S3 data to relational tables using JPA. Store product image metadata in the database while associating physical images with S3 URLs.

Query and Process: Utilize SQL queries and JPA to manipulate and retrieve relational data. Organize hierarchical relationships (e.g., parent-child categories) and join related tables (e.g., Cart with User and Product).

API Integration: Convert relational table data back into JSON format for API responses. Ensure efficient filtering, sorting, and aggregation of data before API delivery.

3) where it's taken from

Primary Data Source: AWS S3 (JSON files and PNG images).

Relational Database: Organized into tables (Product, Cart, Category, User) for efficient storage and processing.

S3Service: Handles fetching raw data from S3 and passing it to backend services for processing.

4) how/why you use it

Frontend API Responses: Convert complex, relational data into simplified JSON for frontend use. Provide optimized APIs for category navigation, product display, cart management, and user data.

Data Organization and Efficiency: Transform unstructured JSON into a structured database schema, enabling efficient querying and sorting. Maintain image metadata and S3 URLs for frontend image delivery without unnecessary duplication.

VI. USE CASES

A. Use Case 1 - Login

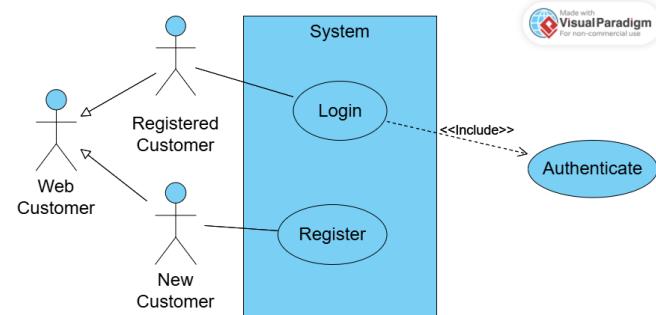


Figure 10: Use Case 1

1) Primary Actor

Web Customer (Registered Customer or New Customer)

2) Stakeholders and Interests

Web Customer: Wants to log into the system or register for an account to access personalized services, such as viewing cart items and enjoying special offers.

System: Aims to securely authenticate customers and allow new customers to register, ensuring personalized experiences.

3) Preconditions

The customer is on the login or registration page of the website. The system has a valid user database for both registered customers and new customer registration.

4) Main Scenario

1. Web Customer accesses the Login page of the system. System presents the login form asking for username and password.
2. Web Customer (if already registered) enters their username and password. System authenticates the credentials by checking the authentication data.
3. If the credentials are valid: System grants the Registered Customer access and redirects them to their personalized dashboard or home page.
4. If the Web Customer is a New Customer: Web Customer can choose to Register a new account. System presents a registration form to input necessary details such as name, email, password, and other relevant information.
5. The New Customer completes the registration form and submits the data. System checks the provided information for validity (e.g., ensures the email is not already used).
6. If registration is successful: System creates a new customer profile and logs in the New Customer, redirecting them to their personalized home page.

5) Special Requirements

The System must ensure security by encrypting passwords and securely handling authentication requests. The registration form should validate customer input, such as email format and password strength. The login and registration process should be responsive, user-friendly, and fast to minimize abandonment.

6) Assumptions

The system has an existing customer database for authentication and account creation. The Web Customer has internet access and is able to use a web browser to interact with the login and registration process.

B. Use Case 2 - Search Product

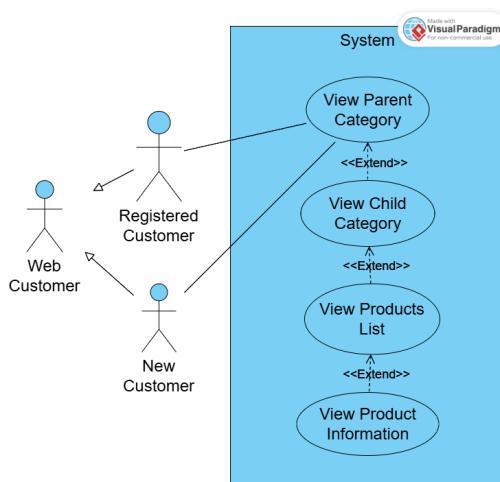


Figure 11: Use Case 2

1) Primary Actor

Web Customer (including both Registered Customer and New Customer)

2) Stakeholders and Interests

Web Customer: Wants to search for products and view detailed information to make an informed purchasing decision.

System: Aims to provide accurate product information and enhance the customer experience.

3) Preconditions

The customer is logged into the website and has permission to view products. The system should have a well-structured category and product catalog to provide relevant information.

4) Main Scenario

1. Web Customer accesses the system and views the parent category page. System provides a list of available parent categories for the customer to browse.
2. Web Customer selects a desired parent category. System displays a list of child categories under the selected parent category.
3. Web Customer selects a desired child category. System displays a list of products available in the selected child category.
4. Web Customer selects a desired product. System displays the detailed product information.
5. Web Customer reviews the product information and makes a purchasing decision.

5) Special Requirement

The Web Customer should be able to view product details clearly, and the UI should be intuitive to help customers easily navigate through categories and products. Product information should include price, detailed description, reviews, and availability.

6) Assumptions

The system maintains an up-to-date product catalog and clearly defines the category structure.

C. Use Case 3 - Cart

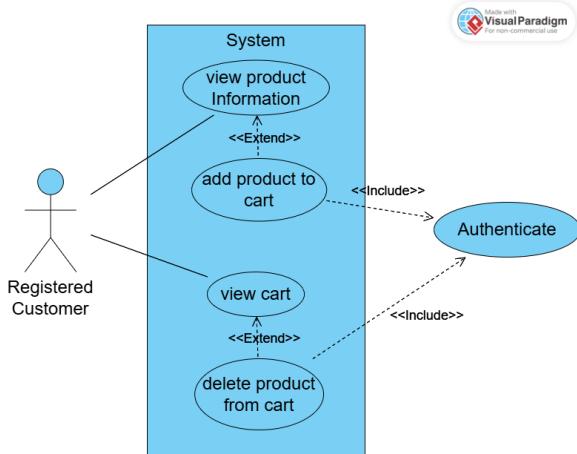


Figure 12: Use Case 3

1) Primary Actor

Registered Customer

2) Stakeholders and Interests

Registered Customer: Wants to view product details, add products to the cart, delete items from the cart, and review the cart contents to make a purchase decision.

System: Aims to display product details, manage the shopping cart, and ensure customers are authenticated to securely perform actions like adding or removing items from the cart.

3) Preconditions

Registered Customer must be logged in and authenticated to perform actions such as adding products to the cart or deleting items from it. The system must have a properly functioning product catalog and shopping cart management system.

4) Main Scenario 1

View Product Information and Add Product to Cart

1. Registered Customer selects a product from the product list and views the product details page. System displays the detailed information of the selected product.

2. Registered Customer decides to add the product to the shopping cart. System provides the Add Product to Cart feature.

3. Upon selecting Add Product to Cart, System prompts the customer to authenticate before adding the product to the cart. Authenticate: The customer is required to log in, and after successful authentication, the product can be added to the shopping cart.

4. After successful authentication, System adds the selected product to the Registered Customer's shopping

cart and either redirects to the cart page or updates the current page to reflect the added product.

5) Main Scenario 2

View Cart and Delete Product from Cart

1. Registered Customer selects the View Cart option to review the items in their cart. System displays the list of products currently in the customer's shopping cart.

2. Registered Customer decides to delete an item from the cart. System provides the option to delete the product from the cart.

3. Upon selecting Delete Product from Cart, System prompts the customer to authenticate before deleting the item. Authenticate: The customer is required to log in, and after successful authentication, the product can be deleted from the cart.

4. After successful authentication, System removes the selected product from the Registered Customer's shopping cart and updates the cart page accordingly.

6) Special Requirements

System must securely handle the Authenticate process to ensure customers' accounts are properly protected during actions like adding or deleting items in the cart. The cart should reflect real-time updates when products are added or removed. Authenticate should be quick and user-friendly to provide a seamless experience for customers.

7) Assumptions

Registered Customer already has an account and must log in to use cart-related features (adding or removing products). System should have a functional shopping cart feature and ensure that product additions and deletions are immediately reflected in the cart.

VII. DISCUSSION

During the project, we faced several challenges. Designing the database structure was complex, as the process of collecting data and creating the Entity-Relationship Diagram (ERD) required three revisions, leading to significant delays in uploading data to the S3 bucket. Implementing the database in Spring was also challenging due to permissions and policy-related issues when retrieving data from S3. Additionally, deploying the backend on an EC2 instance posed difficulties, as the t2.micro instance's low performance and lack of query optimization caused frequent server downtimes, which were temporarily resolved using a swap file. Finally, communication issues within the team, especially due to frequent changes in the frontend's technical feasibility, resulted in constant modifications to the required APIs, delaying their creation and testing. These challenges highlighted the importance of clearer communication and better team alignment to improve efficiency and minimize delays.