ShoppingNova

Group Name: ITCT

1st Hong-Sung Mun
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 that 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 makes 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 prefetching 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

- PostgreSQL

  PostgreSQL is an open-source relational database management system known for its high stability and data integrity. In the shopping mall project, PostgreSQL is used to efficiently store and manage structured data such as products, users, and order information. It supports transaction management and adheres to ACID (Atomicity, Consistency, Isolation, Durability) properties, ensuring data reliability. Optimized for handling large volumes of data, PostgreSQL performs well in large-scale applications.

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. User Account Management

1) Login Page

- Front-end

  Input: Accepts user ID and password input and validates the input values.

  Error Handling: If the user ID or password does not exist in the database, prompts the user to re-enter their credentials.

  DB Query Transmission: Sends the input values as a query to the database to retrieve user information for use in subsequent processes.

- Back-end

  Authentication: Compares the input user ID and password with the stored values in the database and allows login if they match.

  Password Verification: As passwords are stored in encrypted form, the entered password is processed through the same encryption and compared against the stored hash for verification.

2) Sign-Up Page

- Front-end

Input: Collects ID, password, and other personal information from the user, verifying the validity of each field.

Error Handling: Checks for type errors, duplicates, and other input inconsistencies.

- Back-end

Unique User Number Assignment: Automatically generates a unique user number (user_number) for each new user.

ID Duplication Check: Checks if the input user ID (user_id) already exists. If duplicate, returns an error message and halts registration.

Password Security: Encrypts the password for storage, enforcing a minimum length and character combination requirements (e.g., at least 8 characters, including numbers and special characters).

Date of Birth Format Check: Validates the format of the date of birth input and returns an error if there is a format issue.

Email and Phone Number Validation: Validates the format of the email and phone number; if invalid, returns an error message.

Registration Completion Notification: Optionally sends a welcome email or SMS notification to confirm successful registration.

3) User Page

- Front-end

Profile Editing: Provides a page where users can view and edit their profile information.

View Purchase History: Provides a page where users can view their purchase history, with an option to filter by date.

- Back-end

Load Purchase History: Retrieves the user's purchase history from the database.

Profile Editing: Allows users to update personal information, such as name, phone number, and email. Updated information is validated before updating the database.

Identity Verification: Requires identity verification before editing personal information, supporting SMS or email verification based on registration details.

4) User Information Table

user_id: int [pk, increment]
user_name: varchar(255)
email: varchar(255)
password: varchar(255)
grade: int

Table attribute can be changed in development.

5) Order Information Table

order_id: int [pk, increment]
user_id: int [fk]
product_id: int [fk]
quantity: int
total_price: int [KRW]
order_date: datetime

Table attribute can be changed in development.

B. Product Management

1) Product Shopping Page

- Front-end

Product Details Page: When a user clicks on a specific product, a detailed page is displayed, including the product name, price, and description.

- Back-end

New Product Registration: Required fields include product name, price, initial stock quantity, category, brand, and product description. A unique product ID is automatically assigned, with options to link the product to appropriate categories and brands. Administrator can select from existing categories and brands, with the option to add new categories or brands as needed.

Product Information Management: Provides functionality for actions such as adding stock or deleting specific products. Notifies the admin when stock reaches zero and marks the product as out of stock. Allows viewing of detailed information for selected products (name, price, description, image, stock status, etc.).

2) Product Information Table

product_id: int [pk, increment]
product_name: varchar(255)
price: int [KRW]
image_url: varchar(500)
rating: decimal(2,1)
category_id: int [fk]
detail information...

Table attribute can be changed in development.

C. Shopping Cart

- Front-end

Add to Cart: Allows users to add products to their shopping cart via a button.

Checkout: Enables users to purchase all or selected items from their cart on the cart page.

Remove Cart Item: Allows users to remove specific items from their cart.

Adjust Quantity: Adds '+/-' buttons in the cart for users to adjust product quantities.

- Back-end

Add to Cart: Adds the selected product to the user's cart.

Quantity Adjustment: Allows adjusting the quantity of items in the cart using '+/-' buttons, with a maximum limit (e.g., 5 items).

Remove Cart Item: Provides functionality for users to remove specific items from their cart.

Handle Duplicates: Prevents duplicate items by recognizing items already in the cart and not adding them again.

View Cart: Allows users to view items currently in their cart along with the total amount on the cart page.

D. Related Product Recommendation

- Front-end

  Display Recommended Products: Shows similar products in the same category on the product page, allowing easy access to each recommended item.

- Back-end

  Recommended Product Data Processing: Sorts products based on category hierarchy to include models from the same category but with different versions in the recommendation list. For example, if a user views a monitor released in 2020, the backend may recommend a 2024 version as a "latest model" in the recommendations.

E. Product Filtering

- Front-end

  Filter Screen: Provides a filtering screen on the subcategory page where users can adjust filter options such as price, size, and type.

  Apply Filters by Condition: Displays only products that meet the conditions set by the user.

- Back-end

  Filtering Options: Filters products in the database based on conditions such as price range or release year specified by the user. Aggregates and delivers filtered results to the front-end.

  Sorting: Allows users to specify sorting criteria (e.g., price, release year) in ascending or descending order alongside filter conditions. Sorts filtered results on the server before returning sorted data.

  Optimization: Enhances performance of filtering and sorting operations by setting database indexes and optimizing queries.

F. Category Management

1) Category Control

- Front-end

  Category Hierarchy: Displays categories in a hierarchy with main, sub, and sub-sub categories labeled as "Galaxy Cluster," "Galaxy," and "Star Cluster," respectively.

  Category Navigation: When a user selects a top-level category (e.g., Galaxy Cluster), the associated subcategories (e.g., Galaxy) are displayed. Selecting a subcategory reveals the final sub-subcategory or product list.

- Back-end

  Category Navigation: When a user selects Layer 1 (parent category), the back-end returns a list of Layer 2 (child category) items based on the selected category ID. Based on the selection in Layer 2, the final product list is returned.

2) Category Information Table

category_id: int [pk, increment]
category_name: varchar(255)
parent_id: int

Table attribute can be changed in development.

V. Architecture Design & Implementation

A. Overall Architecture

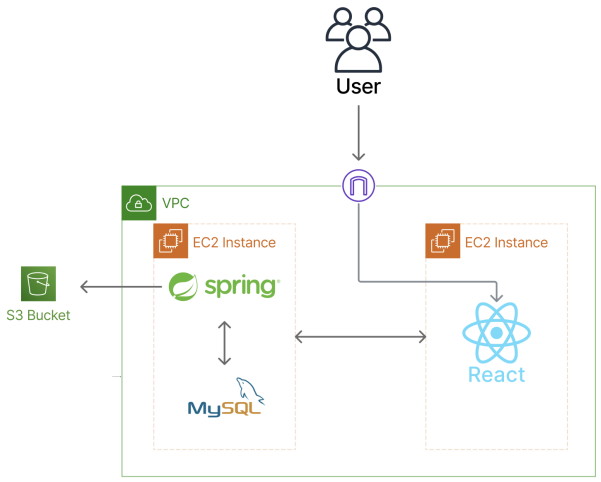

Figure 1: System Architecture

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

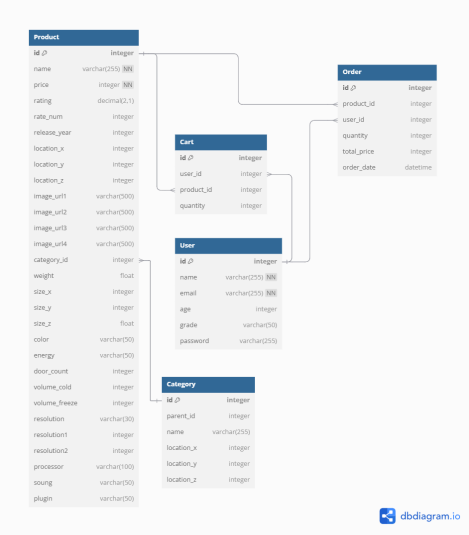

Figure 2: Entity Relationship Diagram

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

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	bspiral.glsl sprial.glsl fragment.glsl glsl.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	Controller
ShoppingNova-BE/ src/java/Entity/Cart	Cart.java CartRepository.java CartService.java	Cart
ShoppingNova-BE/ src/java/Entity/ Category	Category.java CategoryRepository.java CategoryService.java	Category
ShoppingNova-BE/ src/java/Entity/ Product	Product.java ProductRepository.java ProductService.java	Product
ShoppingNova-BE/ src/java/Entity/User	User.java UserRepository.java UserService.java	User
ShoppingNova-BE/ src/java/S3	S3Service.java	S3

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 objectsin 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, Maning 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 themain 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 automates 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: Thisdirectory contains static assets like images and other media files that mightbe used in this app
- ShoppingNova-front/src: This is themain 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 theapplication : e.g., homepage or dashboard.
- ShoppingNova-front/src/components/cart: Components related to the shopping cartview, managing cart items and checkout features.
- ShoppingNova-front/src/components/detail:
  Components for the product detailview, providing a detailed display of individual products.
- ShoppingNova-front/src/components/products:
  Contains components for displayingproduct lists, grids, or galleries.
- ShoppingNova-front/src/components/three:
  Components related to 3D rendering andgalaxy visualization using three.js.
- ShoppingNova-front/src/glsl:Contains WebGL shader files used by three.js for rendering custom graphics oranimations.
  fragment.glsl: The file define pixel-level effects in WebGL rendering. Using assets/particle.png, we define the opacity of pixel.
  bspiral.glsl: The file creates sprial galaxy graphics with tens of thousands of particles.
- ShoppingNova-front/src/routes: Pagecomponents corresponding to different routes in the application.
- ShoppingNova-front/src/utils:Utility functions, constants, and helper modules to simplify repetitive tasksor manage configurations.

- ShoppingNova-front/src/utils/three:
  rand.ts: The file provides functions to randomize the star positoin 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

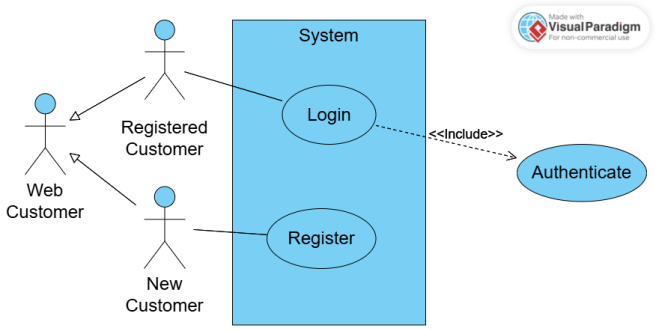

Figure 3: Use Case 1

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

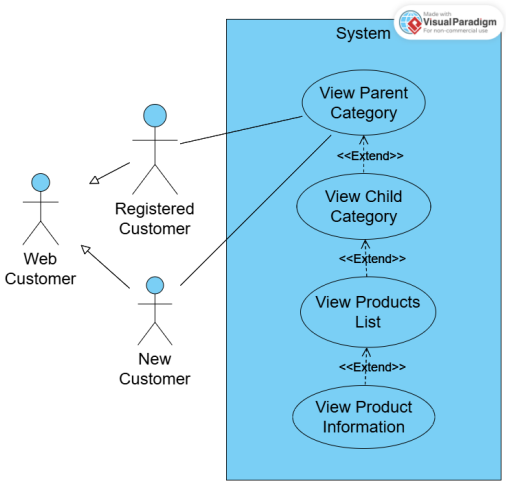

Figure 4: Use Case 2

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

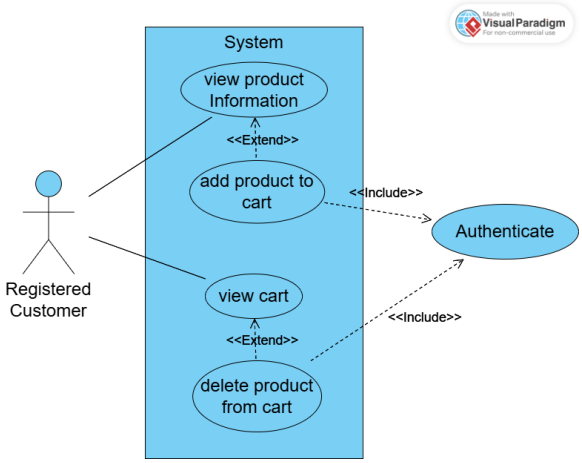

Figure 5: Use Case 3

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, including difficulties in designing the database structure. The process of collecting data and creating the ERD was complex, and we had to revise the ERD three times. This resulted in significant delays in uploading data to the S3 bucket. Additionally, implementing the database in Spring proved challenging due to issues with permissions and policies while pulling data from S3. We also faced difficulties in passing data to the frontend via APIs in JSON format and ensuring proper communication between the backend and frontend. Finally, communication issues within the team, especially with frequent changes in the frontend's technical feasibility, led to frequent changes in required APIs. As a result, the creation and testing of APIs were delayed. These challenges highlighted the importance of clearer communication and better alignment between the teams to improve efficiency and avoid delays.