MadeinFit

Fitment
Software Architecture Document

Version <1.0>

Revision History

Date	Version	Description	Author
26/11/2023	1.0	Introduction	Nguyen Huu Khanh
26/11/2023	1.0	Architecture Goals & Constraints	Nguyen Huu Khanh
26/11/2023	1.0	Use Case Model	Tran Bao Ngoc
26/11/2023	1.0	Package: Model	Le Van Duong
26/11/2023	1.0	Package: UI	Tran Bao Ngoc
26/11/2023	1.0	Package: Controller	Vu Minh Triet
26/11/2023	1.0	Package: Service	Nguyen Phuoc Thinh
26/11/2023	1.0	Package: Middleware	Nguyen Huu Khanh

Table of Contents

Software Architecture Document

1. Introduction

1.1 Purpose

The primary objective of this Software Architecture Document is to provide the foundational architectural choices and principles supporting the Fitment ecommerce website. It functions as a comprehensive guide for stakeholders, developers, designers, and testers engaged in the software development process. By delineating the system's architecture, this document fosters collaboration, minimizes uncertainty, and ensures the development team adheres to a unified perspective and design criteria.

1.2 Scope

In terms of scope, this document encompasses the overarching design of the website, encapsulating its pivotal components, modules, interfaces, and interactions. It provides an in-depth portrayal of the project's structural layout, clarifies the guiding concepts shaping its design, and expounds on the reasons behind the adoption of specific architectural patterns and styles.

1.3 Definitions, acronyms and abbreviations

No	Acronym	Definition
1	MVC	Model - View - Controller
2	UI	User Interface
3	DB	Database
4	API	Application Programming Interface
5	UX	User Experience

1.4 References

1. Software Architecture Document - Course Registration System, Version 1.0, 2001, Wylie College IT.
2. Vision Document.
3. Use-case specification document.

2. Architectural Goals and Constraints

This section outlines the software requirements, goals, and constraints that strongly influence the design of the Fitment website. These goals and constraints are vital in guiding the development team to create a system that meets stakeholders' specific needs. They cover both functional and non-functional aspects, requiring significant attention throughout the software development process.

2.1 Architecture Goals

The architectural objectives encompass key performance criteria that underpin the development of the system:

- Scalability: The system needs to efficiently handle a concurrent load of a minimum of 1000 visits while ensuring consistent and optimal performance.
- Availability: The website dashboard should be accessible to users in Vietnam for at least 95

percent of the business hours every month, ensuring reliable availability.

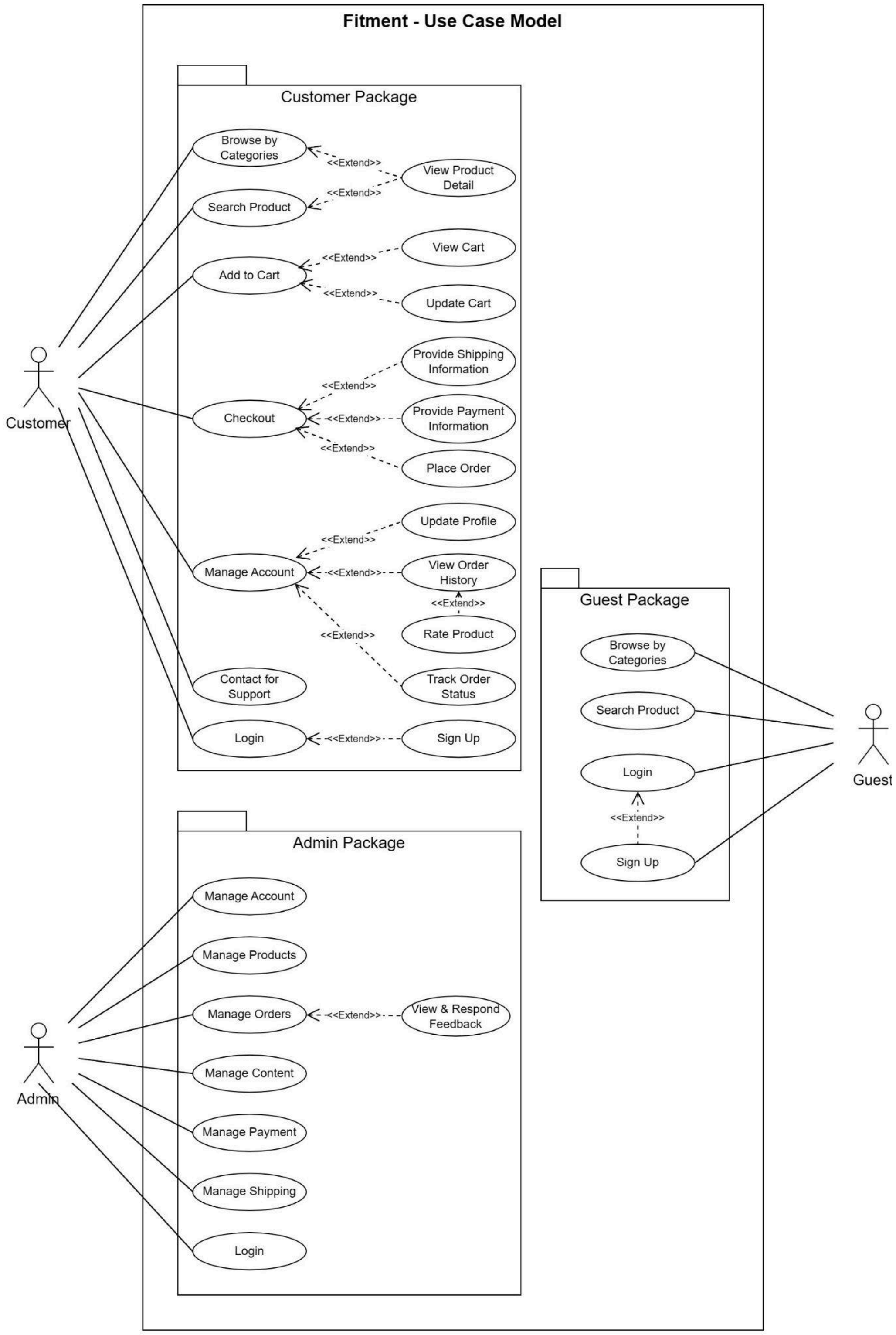
- Reliability: The system should achieve successful performance in at least 80 percent of its defined use cases throughout each month, ensuring dependable functionality.

2.2 Architecture Constraints

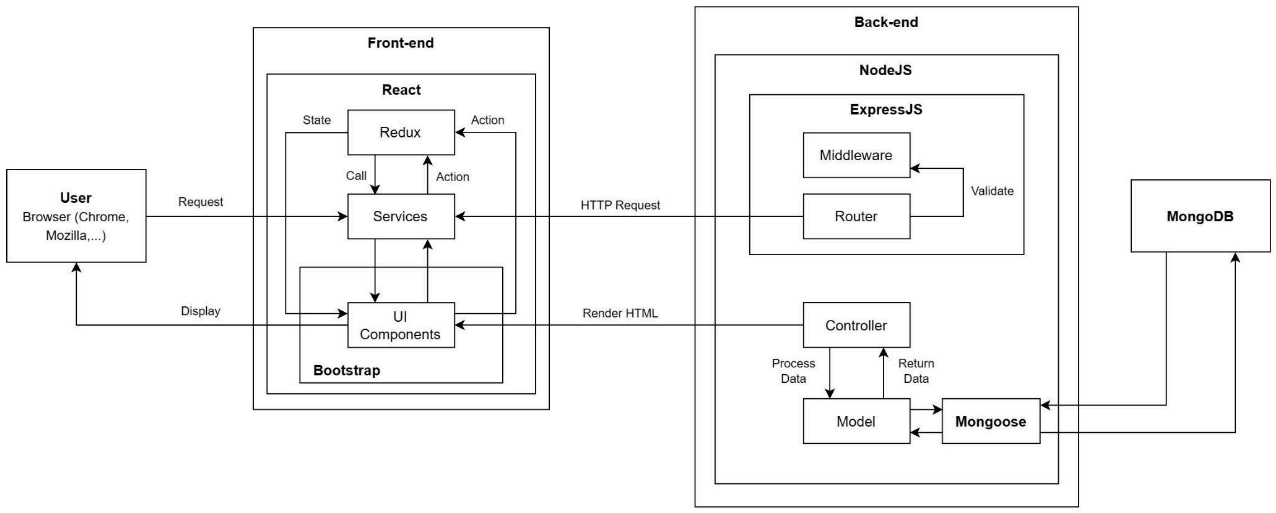
The architecture constraints establish precise directives and limitations guiding the website's development:

- Applicable standards: Support for PCs or laptops operating on Windows 7 or newer versions for optimal accessibility and usage.
- Performance: Optimized response times and efficient resource utilization to ensure rapid loading and smooth user experience, especially during high-traffic periods.
- Programming languages: HTML, CSS, and Javascript are used in building the website.
- Framework: Bootstrap 5, ReactJS, Redux (Front-end), and ExpressJS, Mongoose (Back-end) are employed in the project.
- Usability: Prioritizing user accessibility, the website's interface is designed to be user-friendly and accommodating to individuals with disabilities.
- Data Security: Implementing robust encryption protocols and secure data transmission methods to safeguard user information and prevent unauthorized access or breaches.

3. Use-Case Model



4. Logical View



Description:

We are implementing the MVC architectural pattern in our project. Bootstrap 5 and ReactJS are utilized in the Front-end, while the Back-end leverages NodeJS and the ExpressJS framework. The MVC pattern consists of three elements: models, views, and controllers.

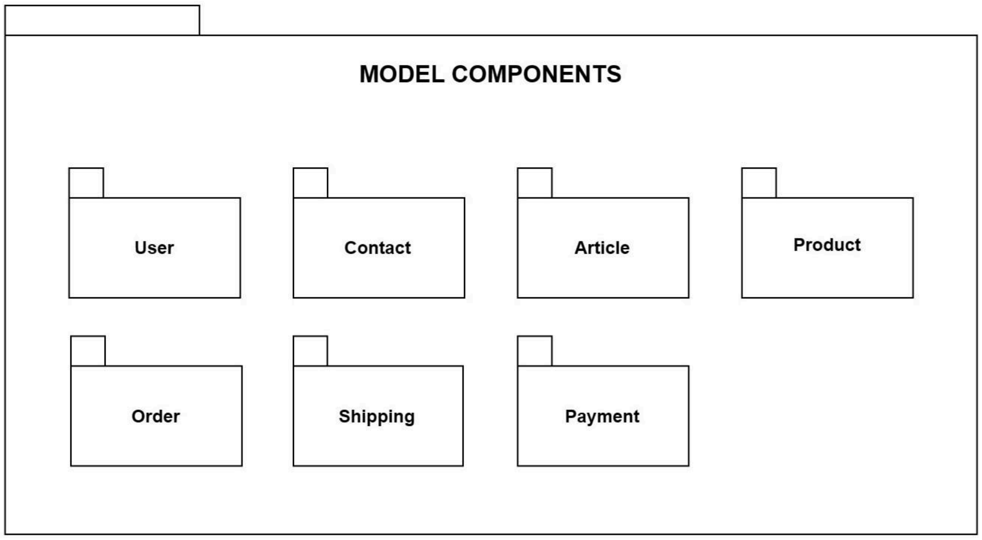
In Front-end:

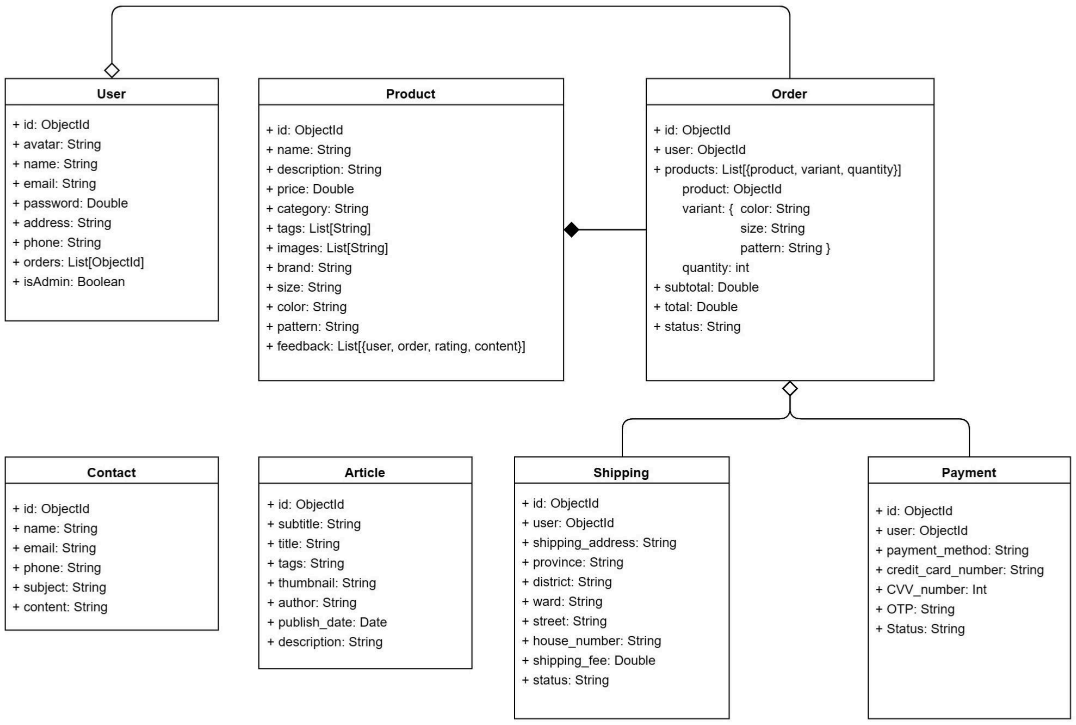
- Redux: A state management tool, stores the application's state and facilitates its access across various components, ensuring consistency and easy management of data.
- Services: These modules handle communication between Front-end and Back-end, managing API calls.
- UI Components: Built using ReactJS, representing the visual elements and user interface.

In Back-end:

- Middleware: A bridge between the request and response cycle, middleware handles various tasks such as data validation, authentication, and error handling.
- Router: Directs requests to respective controllers based on URLs.
- Controller: Intermediaries between models and views. They receive requests from the router, process the necessary data by interacting with models, and send back an appropriate response to the client.
- Model: Represents the application's data structure, defining how data is organized, stored, and manipulated. They interact with the database facilitated by Mongoose to perform CRUD operations, ensuring data integrity and reliability.
- Mongoose: A third-party library integrated into the system, facilitates the management of Create, Read, Update, and Delete (CRUD) operations between the web server and MongoDB server.

4.1 Package: Model





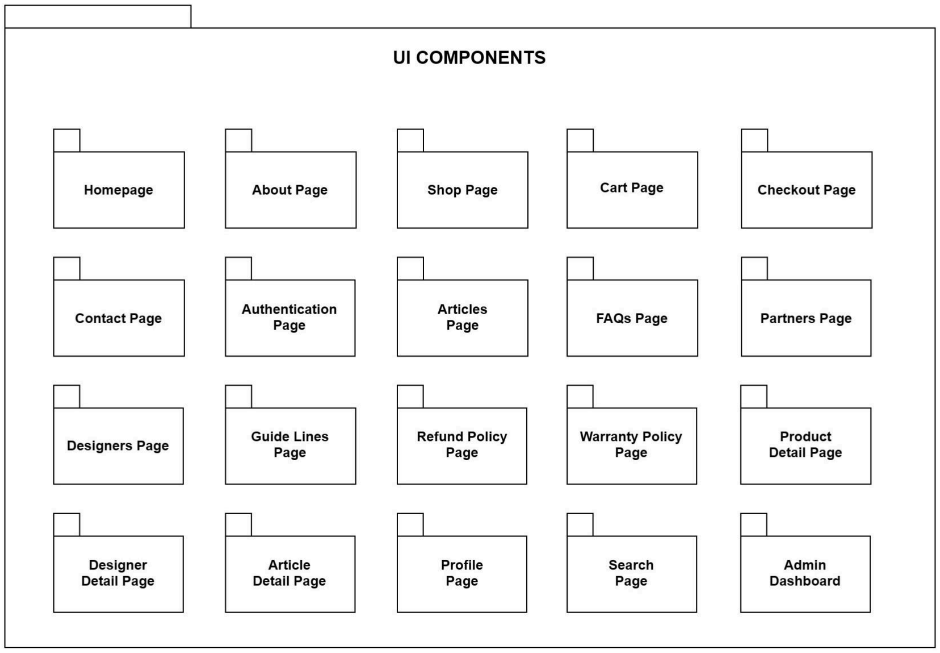
Key classes in Model's package

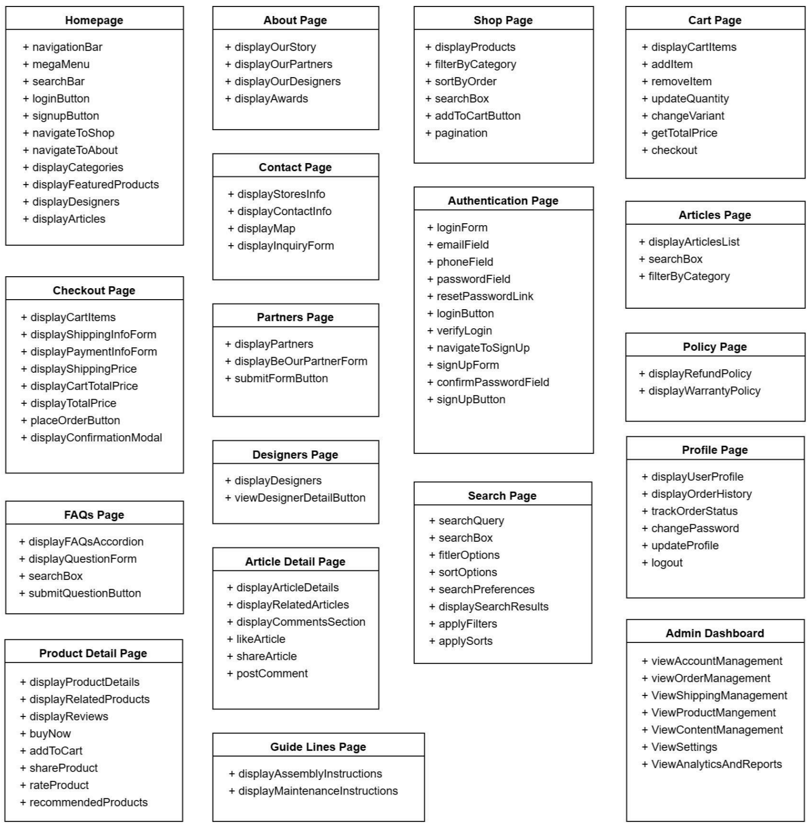
- **Contact:** Storing inquiry's data in Contact page: name, phone, email, inquiry.
- **Article:** Storing article's data like Id, subtitle, title, content, author, publish date.
- **User:** Storing user's data like Id, name, avatar, email, password, etc.
- **Product:** Storing product's data like name, description, price, image, feedback, etc.
- **Order:** Storing order's data like Id, user, subtotal price, total price, status, etc.
- **Payment:** Storing payment's data like user, payment method, credit card number, etc.

- **Shipping:** Storing shipping's data like shipping address, shipping fee, status,etc.

White arrow: *Often represents an implementation or dependency relationship between classes. In this case, the class with the empty triangle arrow refers to an interface or an abstract class that it implements or depends upon.*
Black arrow: *Typically indicates an inheritance relationship between classes. The class with the filled triangle arrow is the subclass, inheriting attributes and methods from the class it points to.*

4.2 Package: UI





Key classes in UI Packages

Homepage:

- Search bar, Navigation bar, Mega menu, Login/Sign Up button: These components are located in the navigation bar respectively (displayed as icon buttons)
- Shop, About, Categories, Feature products, Designers, Articles: Each section contains a brief description (image, content) and a button to navigate to specific pages respectively.

About Page:

- Our story: A brief description about the brand's core values, vision, and mission.
- Our Partners: This section contains logos of partners (manufacturers, sellers, suppliers) that we work with, and a "View more" button.
- Our Designers: This section contains names and images of designers that we work with, and a "View more" button.

Shop Page:

- Displays products in a grid format, showcasing product images, names, prices, and possibly ratings or badges.
- Filter option: Allows users to narrow down the displayed items based on specific criteria such as price range, size, color, brand, or any other relevant attributes.
- Sort option: Enables users to rearrange the order of displayed products based on preferences like price (low to high or high to low), popularity, newest arrivals, or alphabetical order.
- Search box: Offers a direct way for users to look for specific products by entering keywords or phrases.
- Add to cart button: A fundamental component, serves as the primary call-to-action (CTA) that allows users to add items they wish to purchase to the shopping cart.
- Pagination: An essential feature, which divides a number of products into discrete pages, usually with a limited number of items per page.

Cart Page:

- Displays a list of products added to the cart, including details like product names, images, quantities, prices, and possibly attributes (sizes, colors).
- Subtotal and total price: Shows the subtotal, including the cost of all items in the cart, along with shipping fees, or discounts, leading to the total amount to be paid.
- Update quantity: Allows users to modify the quantity of items.
- Continue shopping option: Often includes a button allowing users to return to shopping.
- Checkout button: A CTA to proceed to the checkout process.

Checkout Page:

- Displays a summary of the items in the user's cart.
- Shipping information: Fields for users to enter or confirm their shipping details, including name, address, and contact information.
- Payment information: Fields for users to input billing details, which may include credit card information, billing address, and select payment method.
- Subtotal: Summarizes the total cost of all items in the cart.
- Total: Summarizes the total cost of the order, including taxes, shipping fees, and any discounts applied.
- Shipping options: Presents various shipping methods with associated costs and estimated delivery times. Users can choose the shipping option that best suits their needs.
- Place order button: The primary call-to-action button that users click to confirm and place their order.
- Order confirmation: After successful completion, users receive an order confirmation message or email, including details such as order number, items purchased, and estimated delivery date.

Contact Page:

- Displays a list of stores' information like address, phone number, email, opening hours, Google map, social media link.
- An inquiry form for users to send inquiry/request to contact.

Authentication Page:

- Displays Login or Sign Up Form and Login/Sign up buttons.
- Display Reset password link ("Forgot password?" option) or navigate to the Sign Up page.

Articles Page:

- Displays a list of articles, each article cart contains a thumbnail/preview image, subtitle, title, brief description, and a "Read more" button.
- Filter and Sort options: Provides options to filter articles based on categories, tags, publication dates, or topics. Sorting options may include arranging articles by date, popularity, or relevance.
- Search box: Includes a search bar that allows users to find specific articles by entering keywords or phrases.
- Social sharing buttons, categories/tags tab navigation (optional).

FAQs Page:

- An accordion menu: Gives answers to the most common queries/questions users may have about the product, services, etc.
- A form to collect user's questions.
- Search box: Allows users to quickly find an answer.

Partners and Designers Page:

- Display a list of Partners.
- Display a list of Designers, each link navigates to the Designer detail page.

Policy and Guidelines Page:

- Display paragraphs about privacy policy, warranty policy.
- Display steps to assembly/maintenance/care furniture.

Product Detail Page:

- Describes and showcases a specific product in detail: product images, title, description, price, offer, category, tags, variants and options, availability, inventory, reviews and ratings, "Add to Cart" and "Buy Now" buttons.
- Recommended products list.

Article Detail Page:

- Showcase a specific piece of content in detail, which contains: title and subtitle, author information, publication information, content body, comments section, etc.

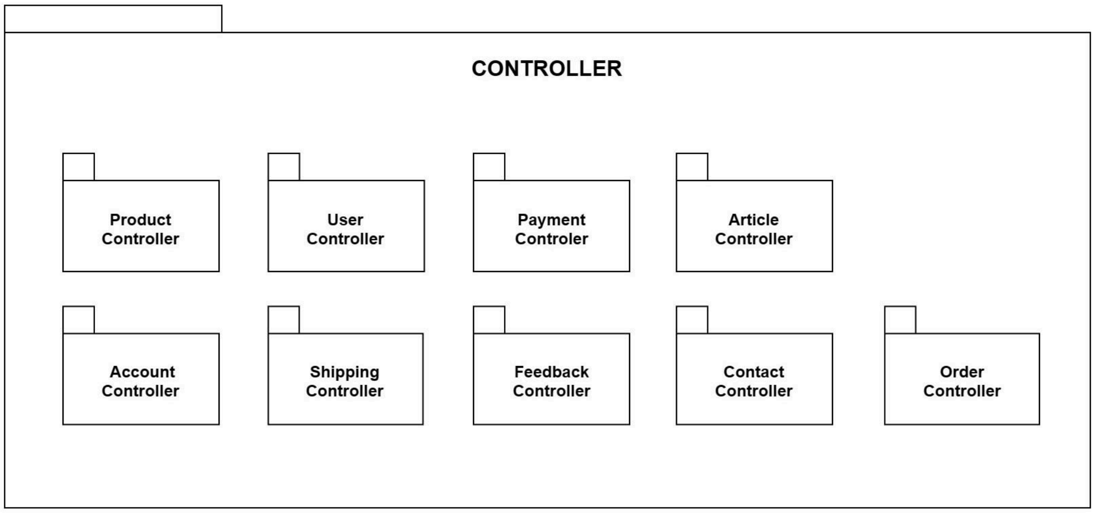
Profile Page:

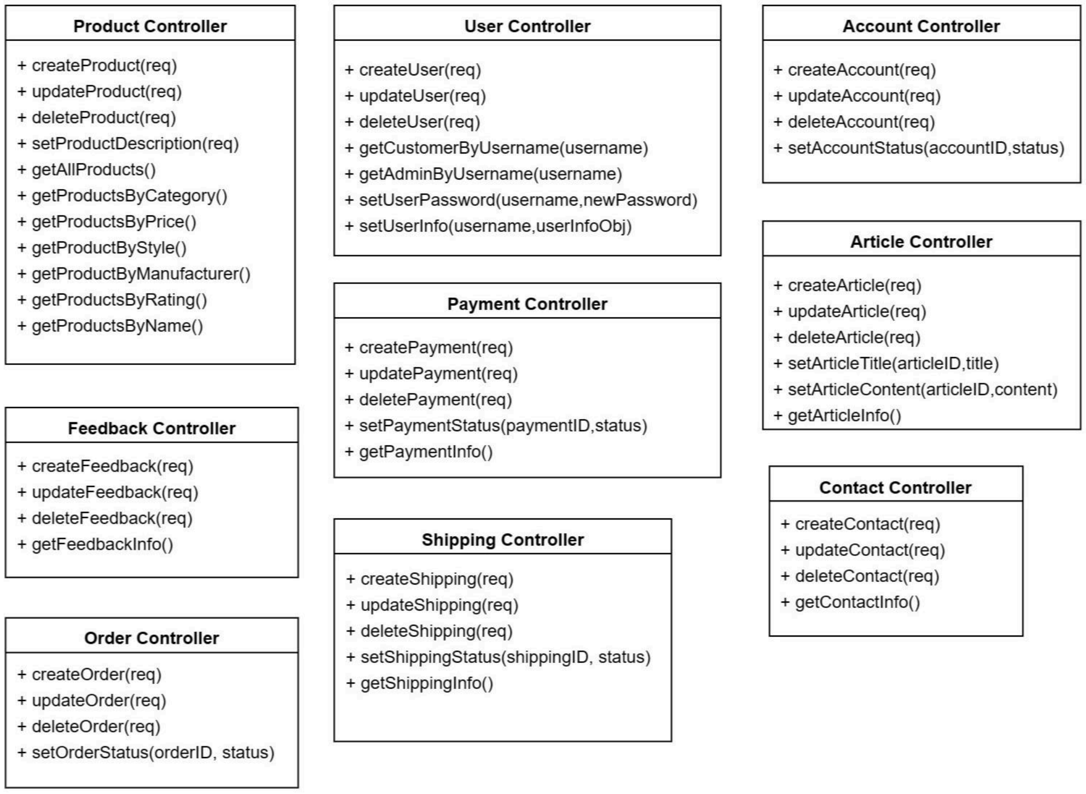
- Serves as a personal space for users to manage their account information, view order history, track order status, change passwords, logout.

Admin Dashboard:

- A central interface designed for administrators or privileged users to manage, monitor, and oversee various aspects of a system.

4.3 Package: Controller



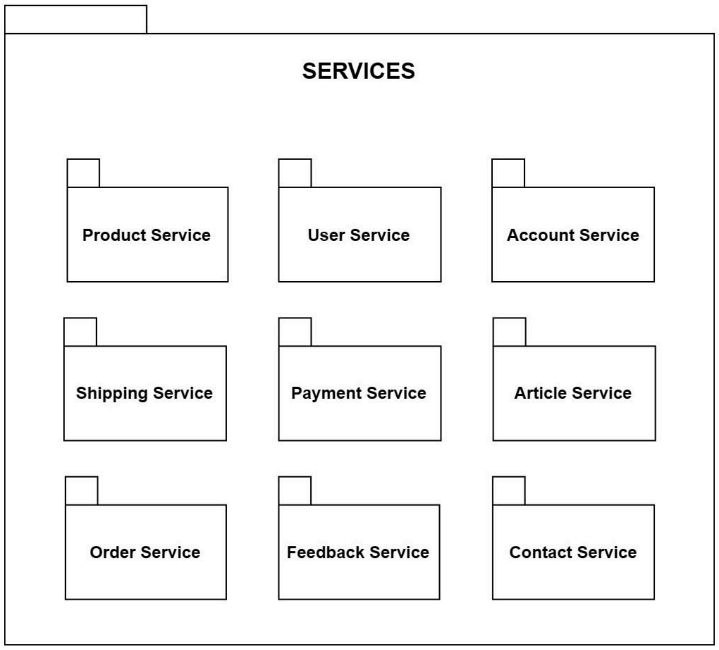


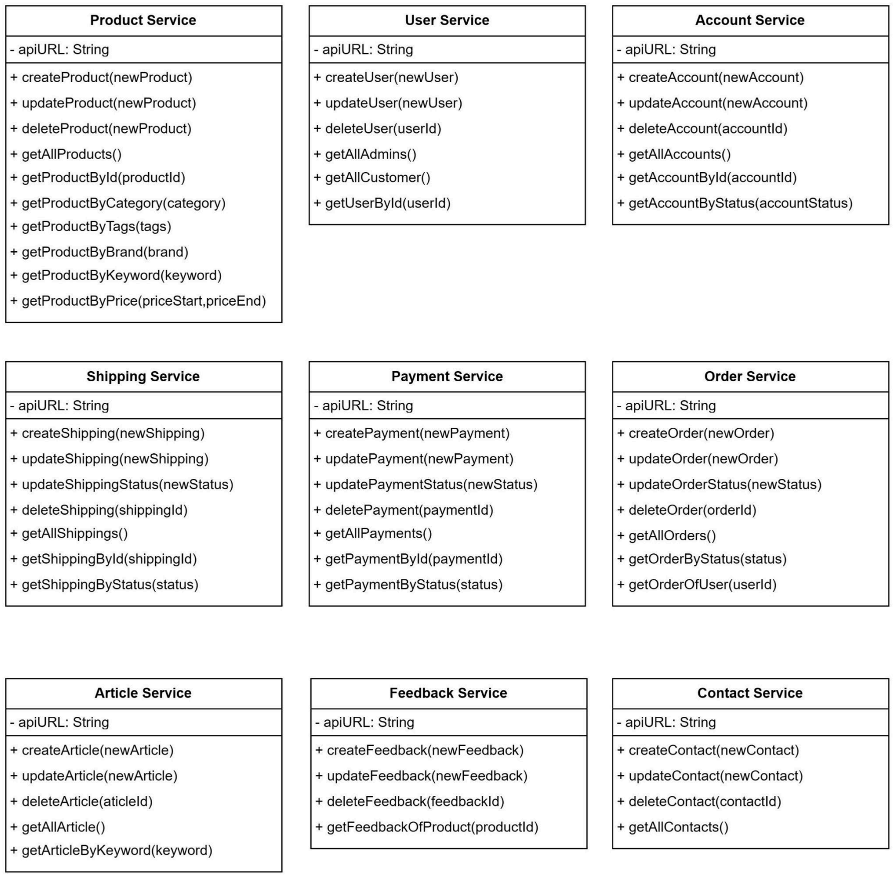
Key classes in Controller:

- **ProductController:** used for filtering products, based on user's choice: by style, by category, by feedback,... Also, ProductController is for getting search results by name.
- **UserController:** involved in the operation of CRUD (Create-Read-Update-Delete) in user account data. Essential for login feature.

- **AccountController:** operates basic CRUD on account management. Also manipulates user's account status.
- **ShippingController:** for shipping service, based on user's choice of shipping for a specified order.
- **PaymentController:** in charge of controlling payment of customer's order.
- **ArticleController:** manages product's description and demonstration.
- **OrderController:** used for performing actions with the order model in the database, OrderController handles customer order feature controlling.
- **FeedbackController:** operates basic CRUD on feedback management.
- **ContactController:** The ContactController plays a crucial role in ensuring that the eCommerce website's contact-related functionalities are efficient, easy to use and develop.

4.4 Package: Service

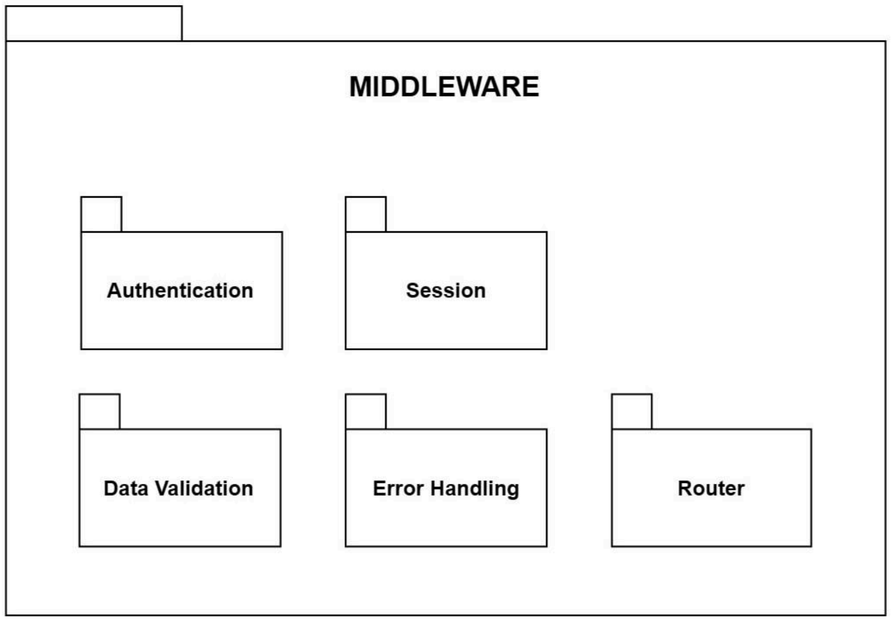


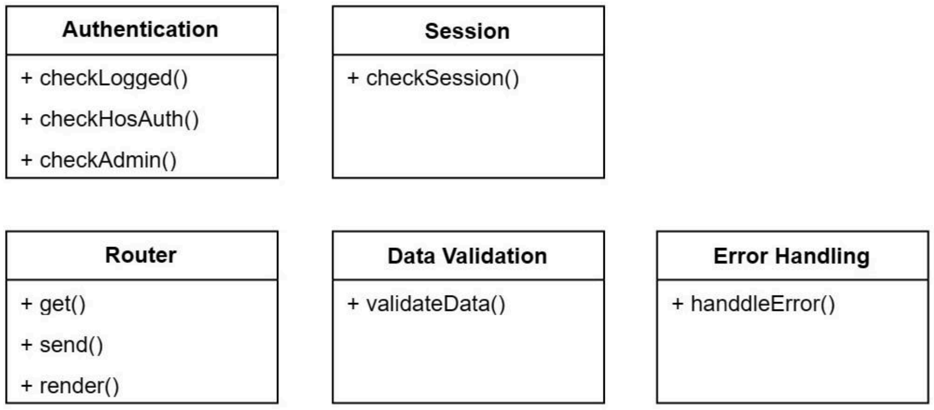

Product Service
- apiURL: String
+ createProduct(newProduct)
+ updateProduct(newProduct)
+ deleteProduct(newProduct)
+ getAllProducts()
+ getProductById(productId)
+ getProductByCategory(category)
+ getProductByTags(tags)
+ getProductByBrand(brand)
+ getProductByKeyword(keyword)
+ getProductByPrice(priceStart,priceEnd)
User Service
- apiURL: String
+ createUser(newUser)
+ updateUser(newUser)
+ deleteUser(userId)
+ getAllAdmins()
+ getAllCustomer()
+ getUserById(userId)
Account Service
- apiURL: String
+ createAccount(newAccount)
+ updateAccount(newAccount)
+ deleteAccount(accountId)
+ getAllAccounts()
+ getAccountById(accountId)
+ getAccountByStatus(accountStatus)
Shipping Service
- apiURL: String
+ createShipping(newShipping)
+ updateShipping(newShipping)
+ updateShippingStatus(newStatus)
+ deleteShipping(shippingId)
+ getAllShippings()
+ getShippingById(shippingId)
+ getShippingByStatus(status)
Payment Service
- apiURL: String
+ createPayment(newPayment)
+ updatePayment(newPayment)
+ updatePaymentStatus(newStatus)
+ deletePayment(paymentId)
+ getAllPayments()
+ getPaymentById(paymentId)
+ getPaymentByStatus(status)
Order Service
- apiURL: String
+ createOrder(newOrder)
+ updateOrder(newOrder)
+ updateOrderStatus(newStatus)
+ deleteOrder(orderId)
+ getAllOrders()
+ getOrderByStatus(status)
+ getOrderOfUser(userId)
Article Service
- apiURL: String
+ createArticle(newArticle)
+ updateArticle(newArticle)
+ deleteArticle(aticleId)
+ getAllArticle()
+ getArticleByKeyword(keyword)
Feedback Service
- apiURL: String
+ createFeedback(newFeedback)
+ updateFeedback(newFeedback)
+ deleteFeedback(feedbackId)
+ getFeedbackOfProduct(productId)
Contact Service
- apiURL: String
+ createContact(newContact)
+ updateContact(newContact)
+ deleteContact(contactId)
+ getAllContacts()

Key classes in services

- **Product service:** The apiURL stores the base url of the product API and contains business logic (create, update, delete, view, search) to interact with the product object or data.
- **User service:** The apiURL stores the base url of the user API and contains business logic (create, update, delete, view, search) to interact with the product object or data
- **Account service:** The apiURL stores the base url of the account API and contains business logic (create, update, delete, view, search) to interact with the account object or data
- **Shipping service:** The apiURL stores the base url of the shipping API and contains business logic (create, update, delete, view, search) to interact with the shipping object or data
- **Payment service:** The apiURL stores the base url of the payment API and contains business logic (create, update, delete, view, search) to interact with the payment object or data
- **Article service:** The apiURL stores the base url of the article API and contains business logic (create, update, delete, view, search) to interact with the article object or data
- **Order service:** The apiURL stores the base url of the order API and contains business logic (create, update, delete, view, search) to interact with the order object or data
- **Feedback service:** The apiURL stores the base url of the feedback API and contains business logic (create, update, delete, view, search) to interact with the feedback object or data
- **Contact service:** The apiURL stores the base url of the contact API and contains business logic (create, update, delete, view) to interact with the contact object or data

4.5 Package: Middleware





Key classes in Middleware

- The **Authentication Middleware** plays a pivotal role in overseeing user authentication and authorization within the system. Its primary function lies in ensuring that only authorized users gain access to specific functionalities and resources of the application, thereby serving as a crucial security layer.
- The **Session Middleware** is tasked with the management of user sessions throughout the project's lifecycle. Its primary role involves facilitating the storage and retrieval of session data associated with individual user interactions within the system.
- The **Data Validation Middleware** is responsible for the thorough validation of incoming data. Its core function revolves around verifying the integrity, accuracy, and compliance of data with predefined validation rules.
- The **Error Handling Middleware** is dedicated to the graceful management of errors that arise during the operation of the website. Its purpose extends to enhancing the application's resilience by effectively

handling errors and offering meaningful feedback to both users and developers when issues occur.

- The **Router Middleware** serves as an intermediary layer between incoming requests and the final processing logic in a web application.

5. Deployment

[Leave this section blank for PA4.

In this section, describe how the system is deployed by mapping the components in Section 4 to machines running them. For example, your mobile app is running on a mobile device (Android, iOS, etc), your server runs all components on the server side including the database]

6. Implementation View

[Leave this section blank for PA4.

In this section, provide folder structures for your code for all components described in Section 4.]