

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

AI4STYLE
Software Architecture Document

Version 1.0

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Revision History

Date	Version	Description	Author
27/11/2025	1.0	First version, defined all needed components in system	All team's members

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Table of Contents

1. Introduction.....	6
2. Architectural Goals and Constraints.....	6
3. Use-Case Model.....	8
4. Logical View.....	9
4.1 Component: View.....	10
Customer Pages.....	10
Admin Pages.....	23
Components.....	24
Hooks.....	25
Services.....	25
2. Homepage.....	25
Components.....	25
Constants.....	26
3. Dashboard.....	26
Components.....	26
Hooks.....	27
Services.....	27
4. Category Management.....	27
Components.....	27
Hooks.....	28
Services.....	28
5. Product Management.....	28
Components.....	28
Hooks.....	29
Services.....	29
6. Staff Management.....	29
Components.....	29
Hooks.....	29
Services.....	30
7. Role Management.....	30
Components.....	30
Hooks.....	30
Services.....	30
4.2 Component: Controllers.....	30
4.2.1 Responsibilities.....	30
4.2.2 Details.....	31

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

4.3 Component: Middleware.....	42
4.3.1 Responsibilities.....	42
4.3.2 Details.....	42
4.4 Component: Business Service.....	45
4.4.1 Responsibilities.....	45
4.4.2 Details.....	46
4.5 Component: Repository.....	59
4.5.1 Responsibilities.....	59
4.5.2 Details.....	59
4.6 Component: Database.....	66
4.7 Component: AI.....	68
4.7.1. ChatbotEngine.....	68
a) Description.....	68
b) Fields.....	68
c) Methods.....	69
4.7.2. LLMClient.....	69
a) Description.....	69
b) Fields.....	69
c) Methods.....	70
4.7.3. ClassifiedAction.....	70
a) Description.....	70
b) Fields.....	70
4.7.4. ClassificationService.....	70
a) Description.....	70
b) Fields.....	71
c) Methods.....	71
4.7.5. ToolRegistry.....	71
a) Description.....	71
b) Fields.....	71
c) Methods.....	71
4.7.6. ToolDefinition.....	71
a) Description.....	71
b) Fields.....	72
4.7.7. VectorStoreGateway.....	72
a) Description.....	72
b) Fields.....	72
c) Methods.....	72

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

4.7.8 ChatRequest.....	72
a) Description.....	72
b) Fields.....	73
4.7.9. ChatResponse.....	73
a) Description.....	73
b) Fields.....	73
4.8 Component: External Service.....	73
4.8.1 LLM (groq).....	73
4.8.2 Momo.....	74
5. Deployment.....	75
6. Implementation View.....	75

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Software Architecture Document

1. Introduction

The purpose of this Software Architecture Document is to describe the high level structure of the AI4STYLE e-commerce fashion system. It provides an architectural overview of the system, including the major components, their responsibilities, and the interactions between them. This document serves as a reference for developers, testers, project managers, and other stakeholders to ensure that the system's implementation aligns with the intended design principles, quality requirements, and technological constraints.

The scope of this document includes architectural decisions related to the system's functional modules, such as product browsing, authentication, ordering, payment, administration dashboard and AI-based features such as Chatbot and Virtual Try-On (VITON). It also defines design constraints derived from the project's schedule, technologies, and non-functional requirements.

This document includes the following sections:

- Purpose and scope of the system's architecture.
- Architectural goals and constraints.
- High-level architectural views.
- Key design decisions and trade-offs.
- Technology stack and external dependencies

This Software Architecture Document is intended to provide a consistent and stable foundation for development throughout all phases of the project.

2. Architectural Goals and Constraints

Architectural Goals:

- Support a modern Client–Server architecture: The system follows a clear separation between the Next.js frontend and the NestJS backend, ensuring scalability and maintainability.
- Provide a smooth and responsive user experience: Page rendering, product browsing, and API responses must feel fast and seamless to users.
- Modular integration of AI components: The Chatbot and Embedding modules (LangChain + LLM) must operate as independent services that can be easily updated or replaced without affecting core features.
- Ensure data security and safe transactions: User authentication, session management, and admin access control must follow secure practices.
- Support extensibility and long-term growth: The architecture should allow easy addition of new modules (e.g., more AI features, advanced search, or analytics).
- Enable rapid development and deployment: The design must be simple enough to implement within the 10-week project schedule while supporting incremental delivery across sprints.
- Use affordable and accessible technologies: Given the zero-budget constraint, the system relies on open-source frameworks and free-tier services.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

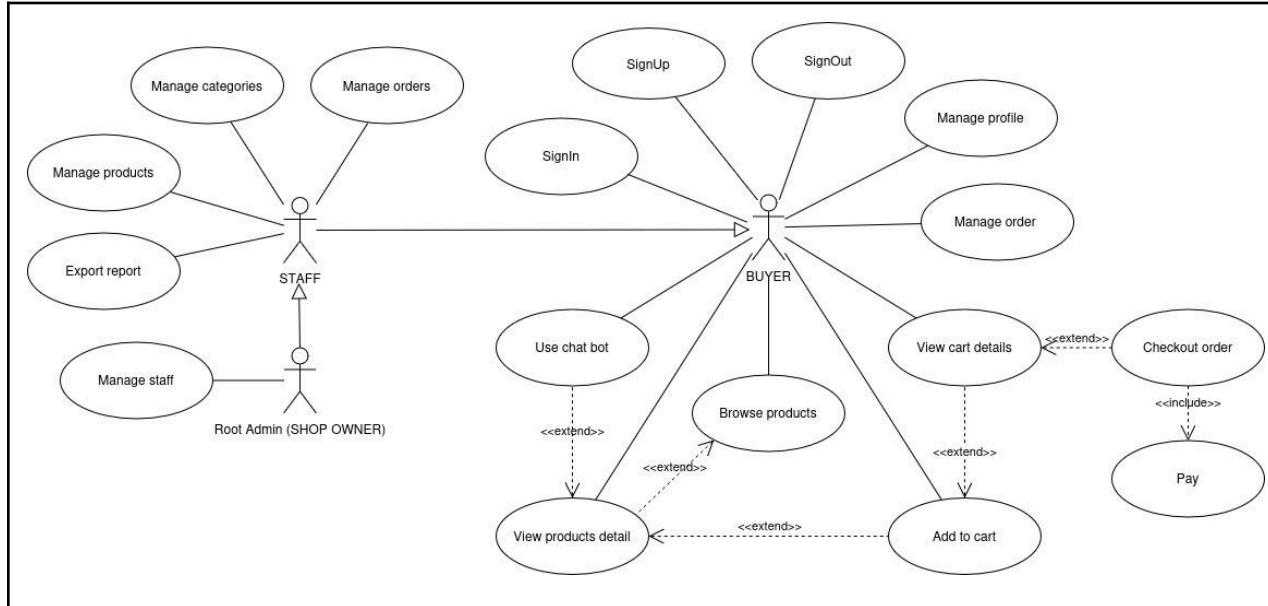
Architectural Constraint:

- Application Environment Constraints:
 - The system is web-only; no mobile-native implementation is in scope.
 - Communication between:
 - Frontend ↔ Backend: HTTP/HTTPS
 - Backend ↔ Supabase DB: TCP/IP
 - Backend ↔ Image Storage: HTTPS
 - Backend ↔ LLM/AI Services: HTTPS
 - Deployment must be compatible with typical free-tier hosting platforms (e.g., Vercel, Railway, Render).
- Technology and Programming Language Constraints:
 - Frontend must use Next.js (React-based framework) for routing, SSR/CSR, and performance optimization.
 - Backend must use NestJS following the Onion Architecture as shown in the diagram.
 - TypeScript is the required programming language across both frontend and backend.
 - AI components must use LangChain for LLM orchestration and vector retrieval.
 - Supabase (PostgreSQL + Vector DB) must be used as the primary database solution.
- Security Constraints:
 - Authentication must be implemented using JWT or a secure session-based mechanism.
 - Passwords must be hashed (bcrypt or equivalent).
 - Role-based authorization is required to separate customer, staff, and administrator privileges.
 - API keys must not be exposed in any public repository.
 - Data transmission should be encrypted using HTTPS when deployed.
- Performance Constraints:
 - API response time for core operations (product listing, cart, orders) should ideally be < 300ms.
 - Chatbot performance must remain responsive by relying on embedding-based retrieval instead of direct LLM calls when possible.
 - Images must be handled through a dedicated Image Storage service to reduce backend load.
 - The system should avoid overly complex architectures (e.g., microservices), due to time and manpower limitations.
- Resource and Cost Constraints:
 - As a zero-budget project, the system must rely ONLY on:
 - Supabase Free Tier.
 - Free or open-source frameworks (Next.js, NestJS, LangChain)
 - Free-tier hosting services
 - No licensed or paid proprietary technologies are allowed.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

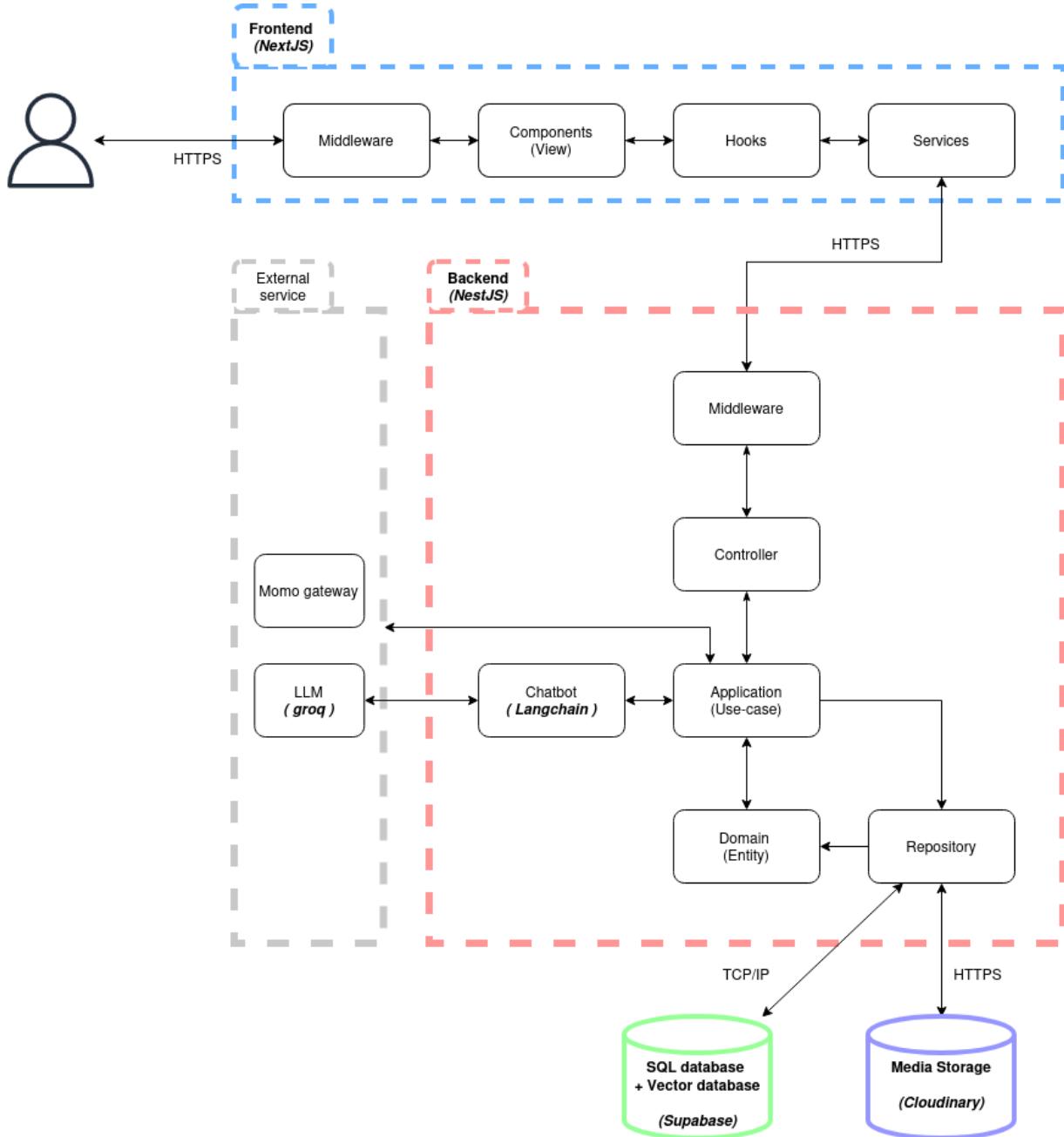
- Team and Schedule Constraints:
 - With a 5-member team, the architecture must enable clear task separation:
 - Frontend development (Next.js)
 - Backend development (NestJS)
 - AI modules (LangChain + LLM)
 - Database/Supabase
 - Given the 10-week schedule and 2-week sprints, each module must be loosely coupled and deployable independently.
- Data and Storage Constraints:
 - Supabase PostgreSQL must store all core data: users, staff, products, orders, etc.
 - Supabase Vector Database must store embeddings for AI retrieval.
 - Product images and VITON-related images must be stored in an external Image Storage service (e.g., Supabase Storage).
 - Secrets and configuration values must be managed securely (environment variables).

3. Use-Case Model



AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

4. Logical View



The system follows an N-Tier Architecture, decomposed into three primary subsystems: Frontend, Backend, and Storage, along with an interface for External Services. Each subsystem is built from distinct, modular components that handle specific responsibilities.

Frontend:

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **View component:** it is responsible for the user interface. It internally layers UI Rendering, State Management Hooks, and API Services to deliver a responsive and interactive user interface.

Backend:

- **Controller Component:** The entry point that orchestrates HTTP requests and validates input.
- **Service Component:** The core component containing business rules and transaction logic, interacts with external service through interfaces..
- **Repository Component:** The data access component that abstracts database interactions.

Storage:

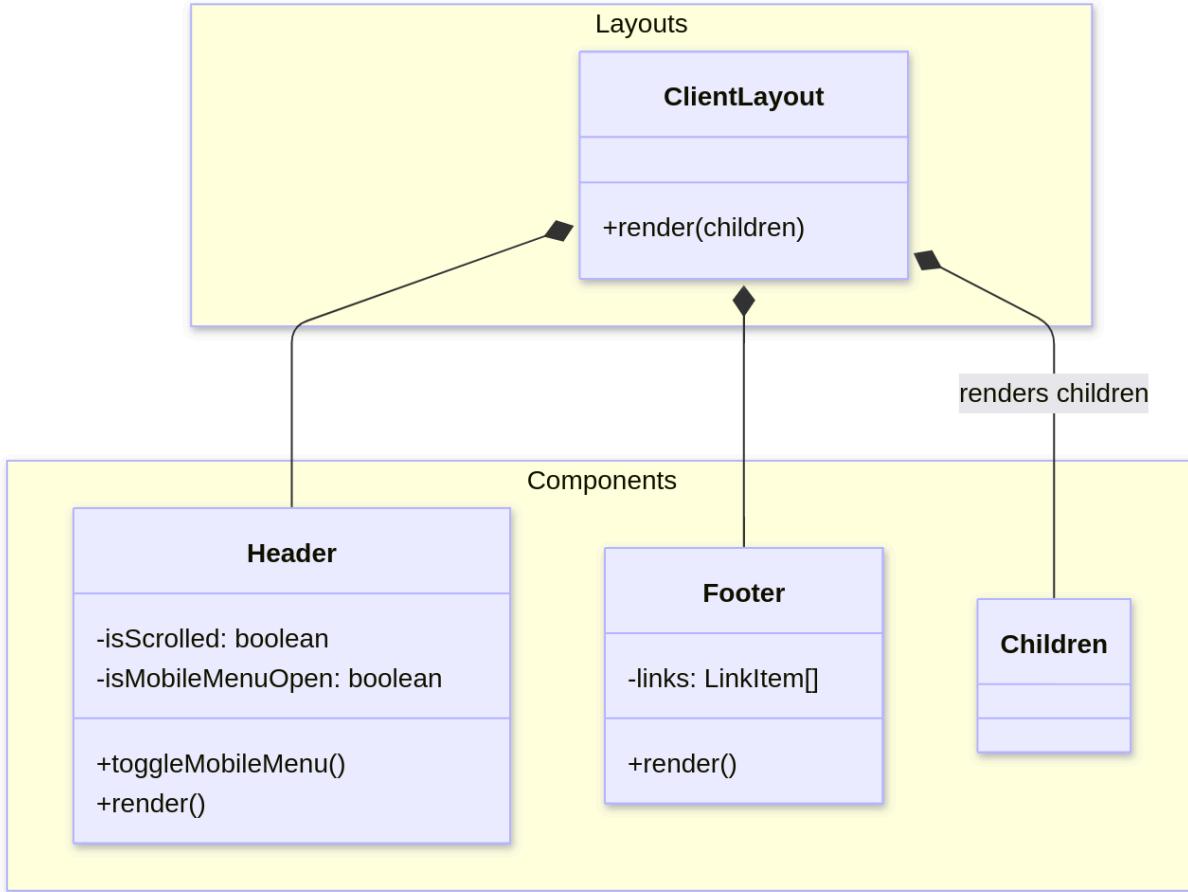
- **Database Component:** A relational database storing structured transactional data (Users, Orders).
- **Media Storage Component:** Object storage for unstructured assets like product images and banners.

4.1 Component: View

Customer Pages

1. Layout

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

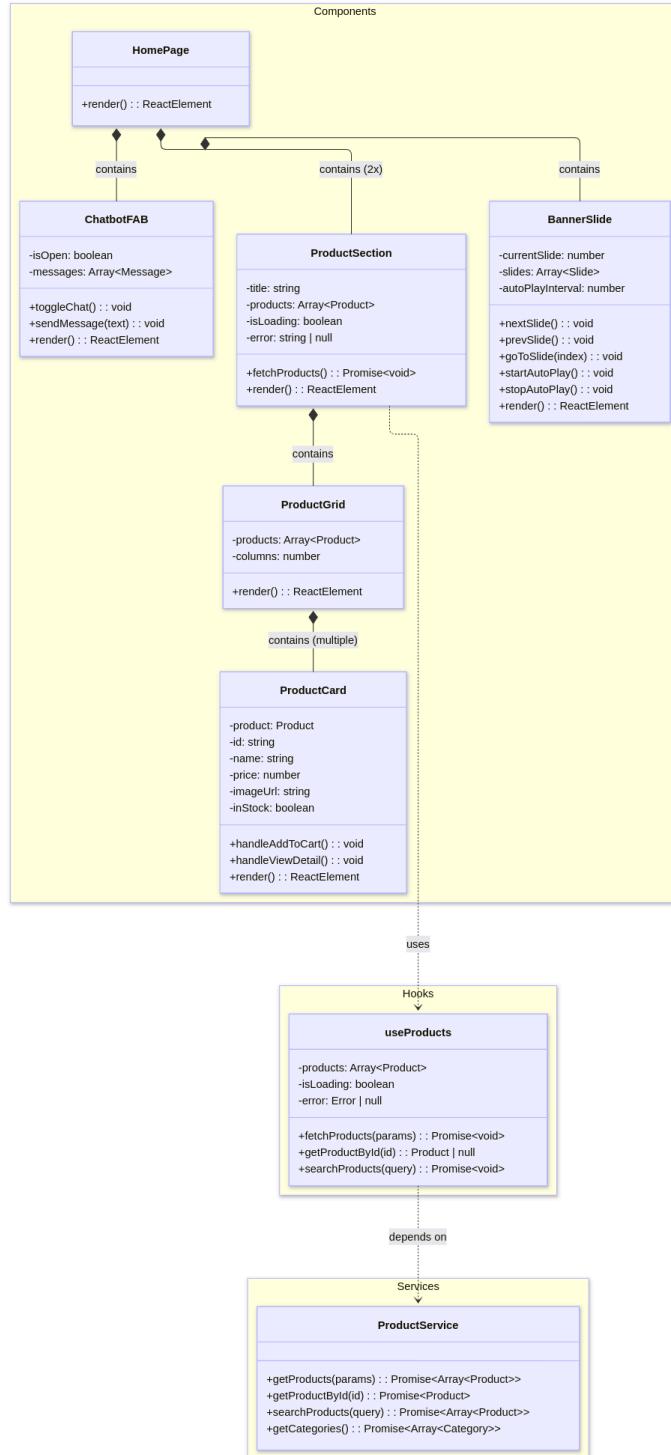


Components

- **ClientLayout**: The top-level wrapper component that renders the children within a consistent site structure (Header and Footer).
- **Header**: The main navigation bar, handling responsive states (mobile menu) and scroll behavior.
- **Footer**: The bottom section of the site contains links and legal information.

2. Homepage

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Components

- **HomePage**: The main landing page view orchestrating the layout of banners, product sections, and features.
- **BannerSlide**: A carousel component that rotates through promotional slides (images/text) with auto-play functionality.
- **ProductSection**: A container component that displays a titled section of products (e.g., "New Arrivals") and handles its own data fetching.
- **ProductGrid**: A layout component that arranges product cards in a responsive grid.
- **ProductCard**: A presentation component representing a single product summary, including the image, price, and "Add to Cart" action.
- **ChatbotFAB**: A Floating Action Button (FAB) that toggles the chatbot interface for customer support.

Hooks

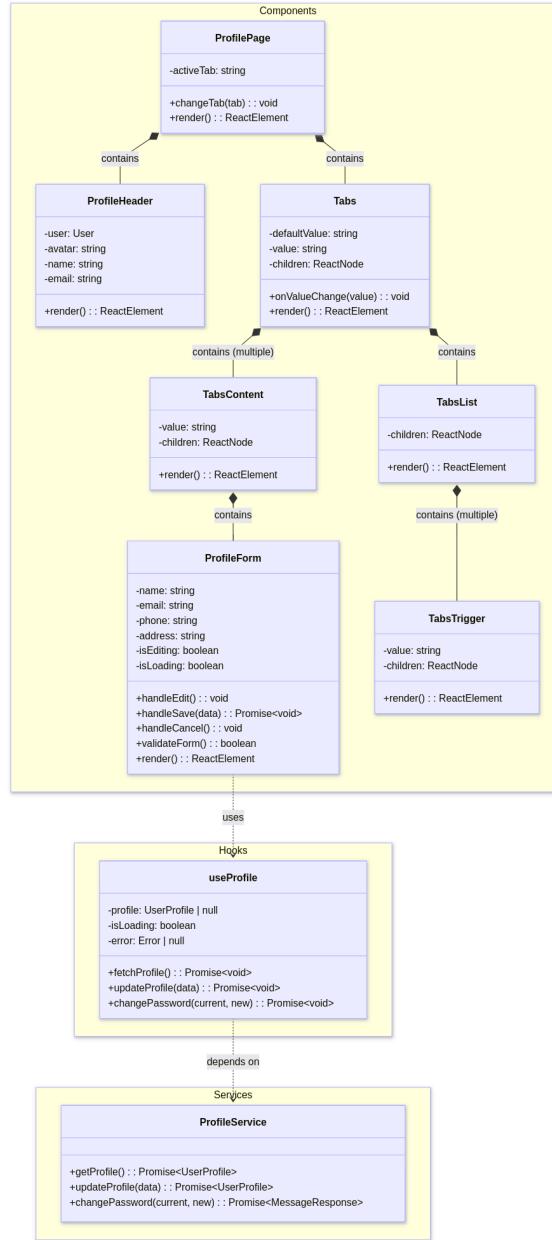
- **useProducts**: Manages the fetching and search logic for products, exposing loading states and the product list.

Services

- **ProductService**: The API service responsible for fetching product lists, details, and categories from the backend.

3. Profile

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Components

- **ProfilePage**: The main container for the user profile, managing the active tab state (e.g., "Info", "Password").
- **ProfileHeader**: A summary component displaying the user's avatar, name, and email at the top of the profile section.
- **ProfileForm**: An editable form allowing users to update their personal details (name, phone, address).

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **Tabs:** A compound component system (Root, List, Trigger, Content) used to organize profile sections.

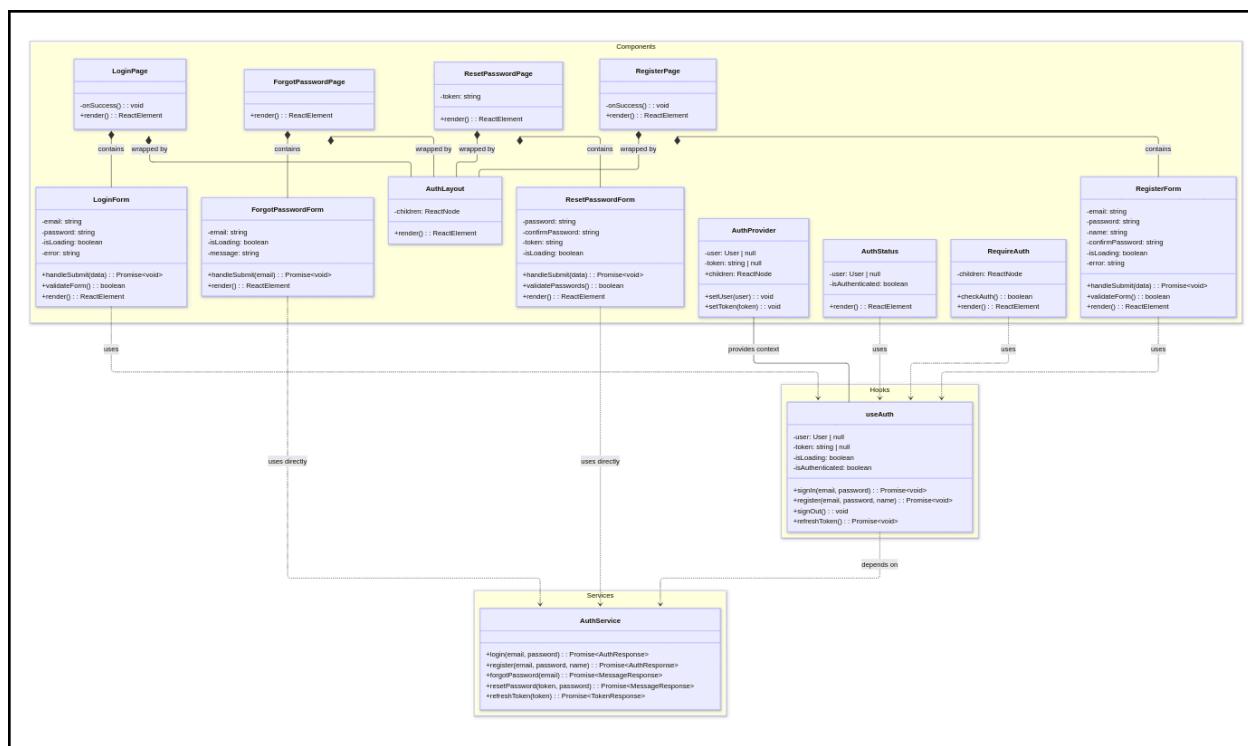
Hooks

- **useProfile**: Manages the fetching and updating of the user's profile data and password changes.

Services

- **ProfileService**: The API service responsible for CRUD operations on the user's own profile data.

3. SignIn/SignUp



Components

- **AuthLayout**: A wrapper component providing a consistent look and feel for all authentication-related pages (Login, Register, Forgot Password).
 - **LoginPage**: The entry point page for existing users to sign in.
 - **LoginForm**: A form component responsible for capturing email/password and triggering the login action.
 - **RegisterPage**: The entry point page for new users to create an account.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **RegisterForm**: A form component for capturing new user details (name, email, password) with validation.
- **ForgotPasswordPage**: The page allowing users to request a password reset link.
- **ForgotPasswordForm**: A simple form to input the email address for recovery.
- **ResetPasswordPage**: The page where users land after clicking the recovery link to set a new password.
- **ResetPasswordForm**: A form component to capture and validate the new password.
- **AuthStatus**: A UI component that conditionally renders user info or login buttons based on the current authentication state.
- **AuthProvider**: A context provider that initializes and maintains the user's session state globally.
- **RequireAuth**: A high-order component (HOC) or wrapper that protects routes, redirecting unauthenticated users to the login page.

Hooks

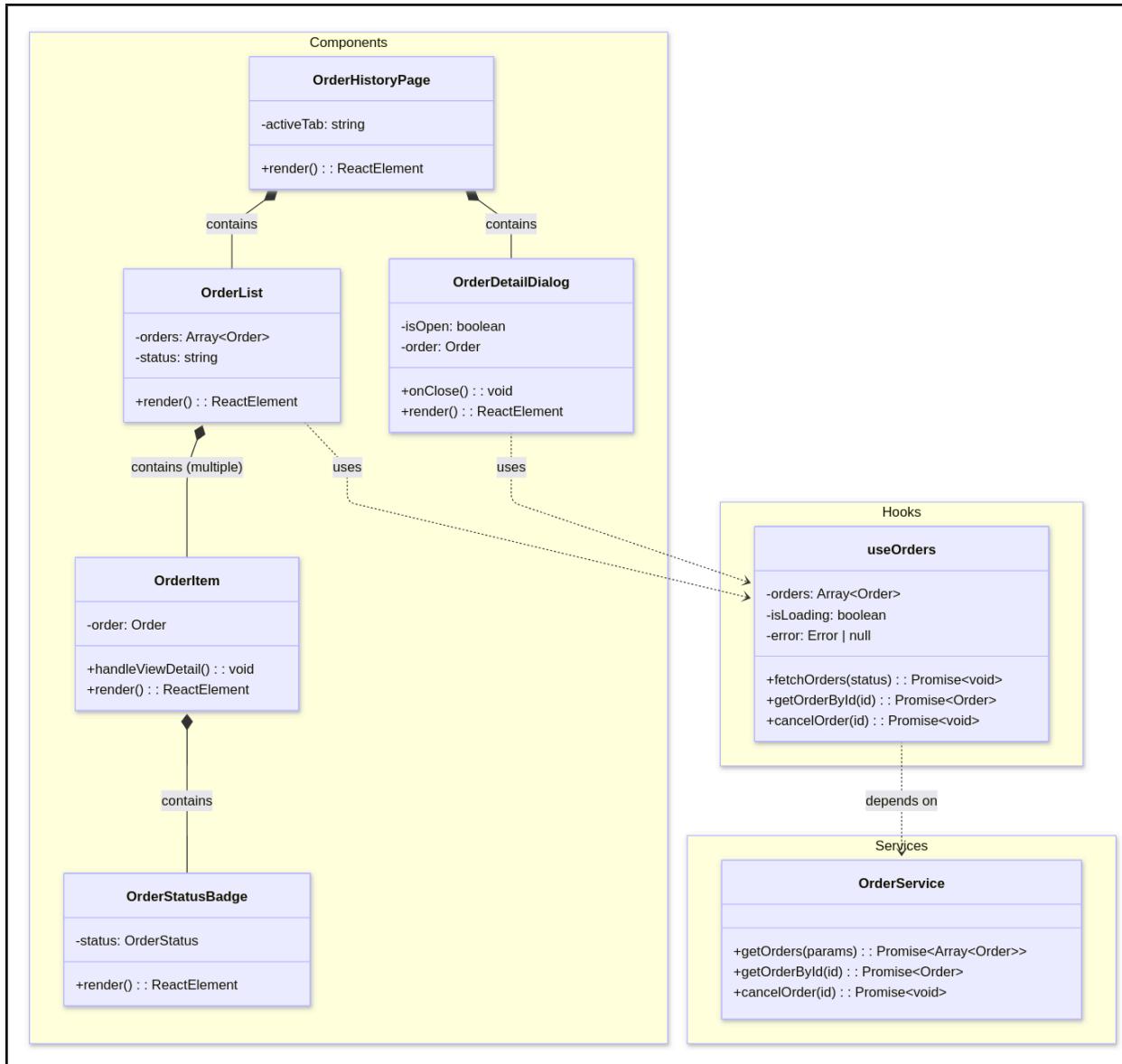
- **useAuth**: A custom hook exposing the current user state (`user`, `token`, `isAuthenticated`) and authentication methods (`signIn`, `register`, `signOut`).

Services

- **AuthService**: The API service responsible for communicating with backend auth endpoints, including login, registration, and token management.

4. Order history

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Components

- **OrderHistoryPage**: The main page listing the user's order history, typically filtered by status tabs.
- **OrderList**: A list component that renders the collection of order items.
- **OrderItem**: A summary component for a single order row, displaying the ID, total, and status.
- **OrderDetailDialog**: A modal component showing the full details of a specific order (products, shipping info) when clicked.
- **OrderStatusBadge**: A visual component displaying the current status of an order (e.g., "Pending", "Delivered") with appropriate styling.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

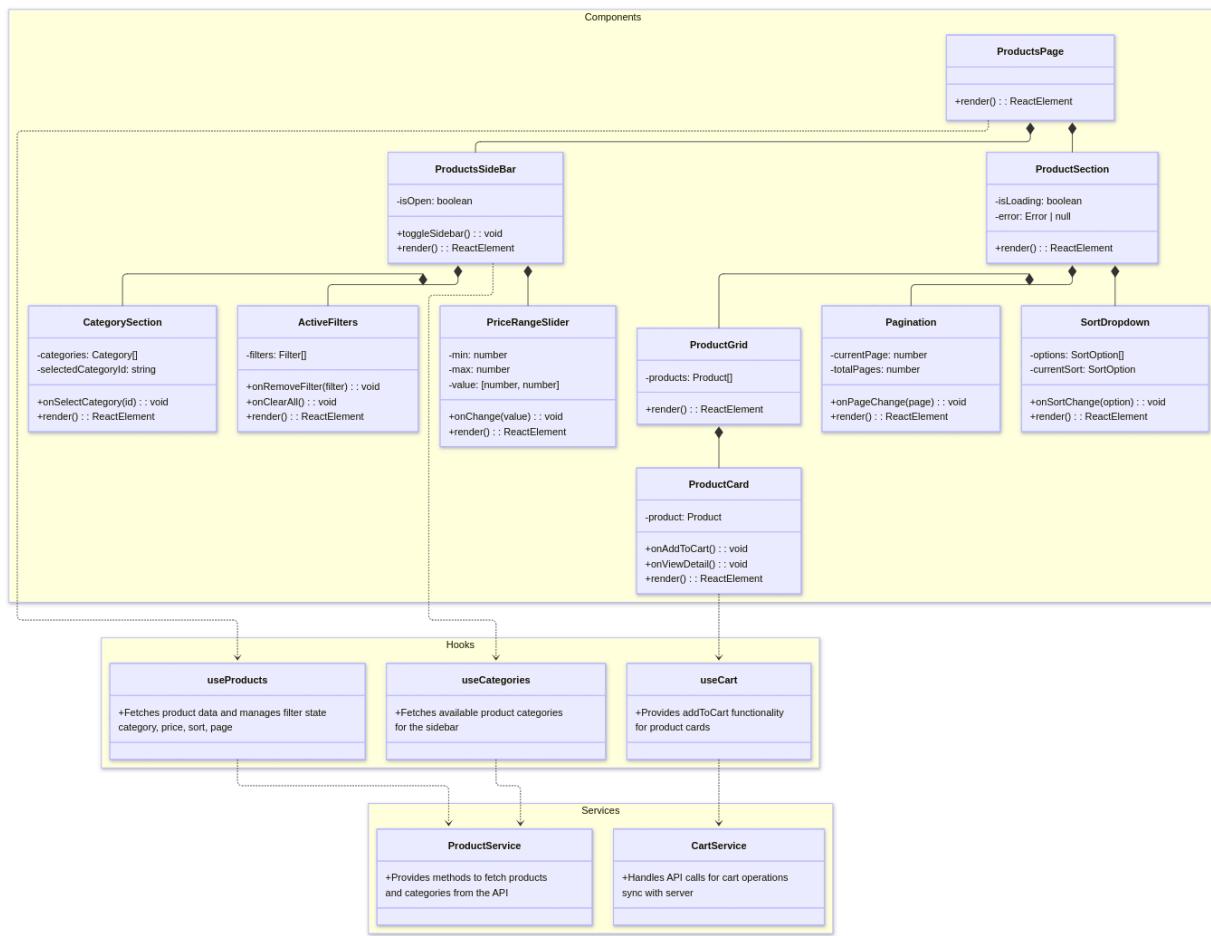
Hooks

- **useOrders**: Manages the fetching of the user's order list and specific order details.

Services

- **OrderService**: The API service responsible for fetching order history and details from the backend.

4. Product list



Components

- **ProductsPage**: Main layout containing the sidebar and product grid.
- **ProductsSidebar**: Displays all filters including categories, price range, and attributes.
- **ProductSection**: Manages the product grid's loading, error states, and display.
- **CategorySection**: Lists all available product categories for filtering.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **ActiveFilters:** Shows currently active filters with options to clear them.
- **PriceRangeSlider:** Allows users to filter products based on price.
- **ProductGrid:** Renders a grid of ProductCard components.
- **Pagination:** Handles navigation across pages of results.
- **SortDropdown:** Provides sorting options such as newest, price, popularity.
- **ProductCard:** Displays individual product info with a link to the detail page.

Hooks

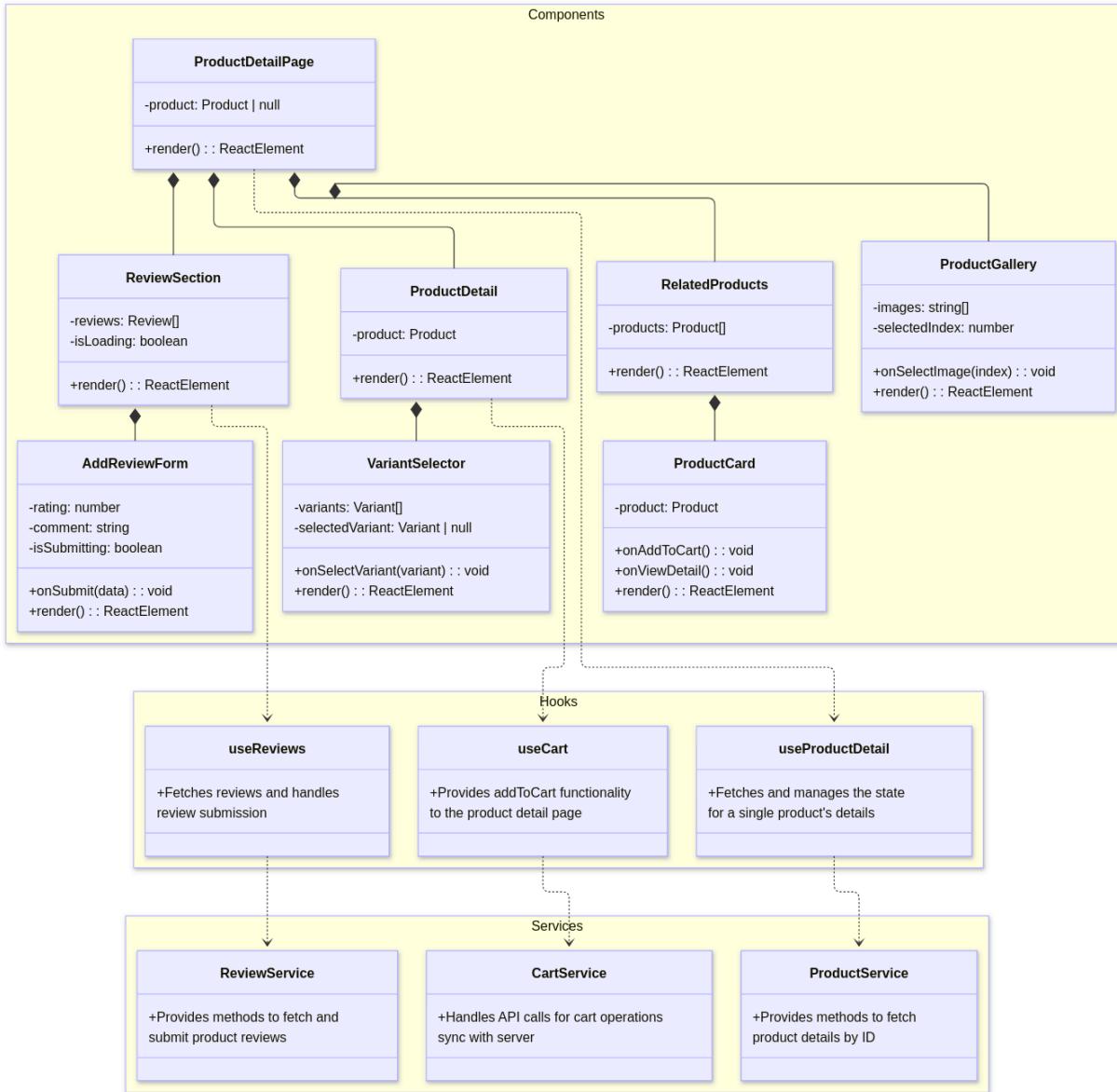
- **useProducts:** Fetches products and manages filter state (category, price, sort, page).
- **useCategories:** Fetches available categories to populate the sidebar.
- **useCart:** Provides add-to-cart functionality for product cards.

Services

- **ProductService:** Fetches product lists and categories from the API.
- **CartService:** Manages server-synced cart operations.

5. Product details

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Components

- **ProductDetailPage**: Main layout for the product detail view.
- **ReviewSection**: Displays user reviews and ratings for the product.
- **ProductDetails**: Shows full product information, including description, specifications, and price.
- **RelatedProducts**: Shows other products similar or relevant to the current product.
- **ProductGallery**: Displays product images in carousel or grid form, with zoom features.
- **AddReviewForm**: Form for submitting a new review and rating.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **VariantSelector:** Allows users to choose product variants (size, color).
- **ProductCard:** Shows basic product info (image, price, name) and links to other product pages.

Hooks

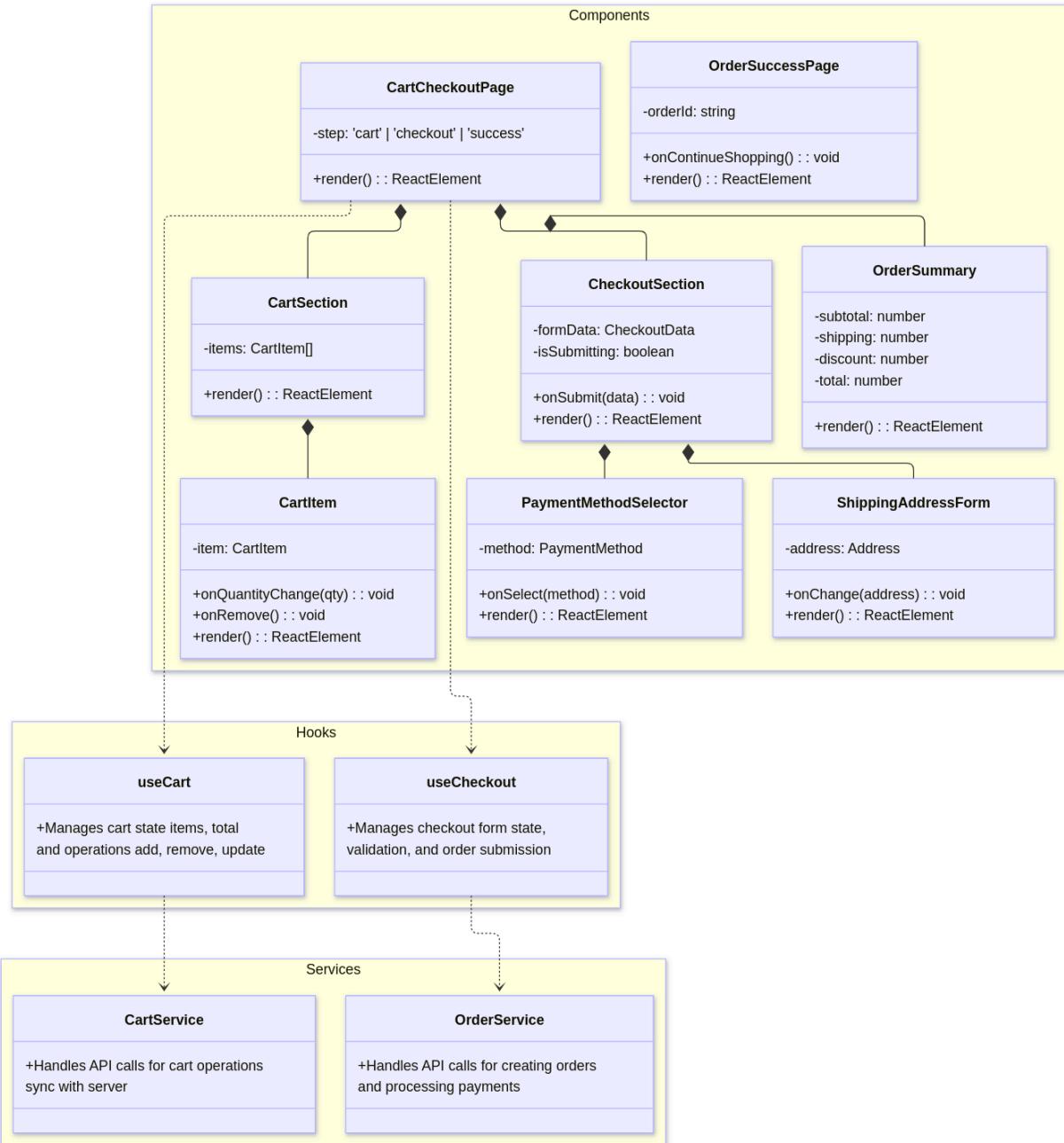
- **useReviews:** Fetches product reviews and manages review submission.
- **useCart:** Provides add-to-cart functionality inside the product detail page.
- **useProductDetails:** Fetches and manages the state of an individual product's data.

Services

- **ReviewService:** Provides API methods for fetching and submitting reviews.
- **CartService:** Synchronizes cart operations with the server.
- **ProductService:** Fetches product details using the product ID.

6. Cart & Checkout

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Components

- **CartCheckoutPage:** Renders the combined cart and checkout experience.
- **OrderSuccessPage:** Shows a confirmation message after the order is successfully placed.
- **CheckoutSection:** Displays the shipping and payment forms.
- **OrderSummary:** Shows the total cost including taxes and shipping.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **CartSection:** Lists all cart items with options to update or remove them.
- **PaymentMethodSelect:** Allows the user to choose a payment option (Card, PayPal, etc.).
- **ShippingAddressForm:** Collects the user's shipping information.
- **CartItem:** Displays individual cart item details, pricing, and quantity controls.
- **CouponInput:** Lets the user enter a discount code to apply to their order.

Hooks

- **useCart:** Manages the cart state including items, totals, and operations like add, remove, and update.
- **useCheckout:** Manages checkout form state, form validation, and triggers order submission.

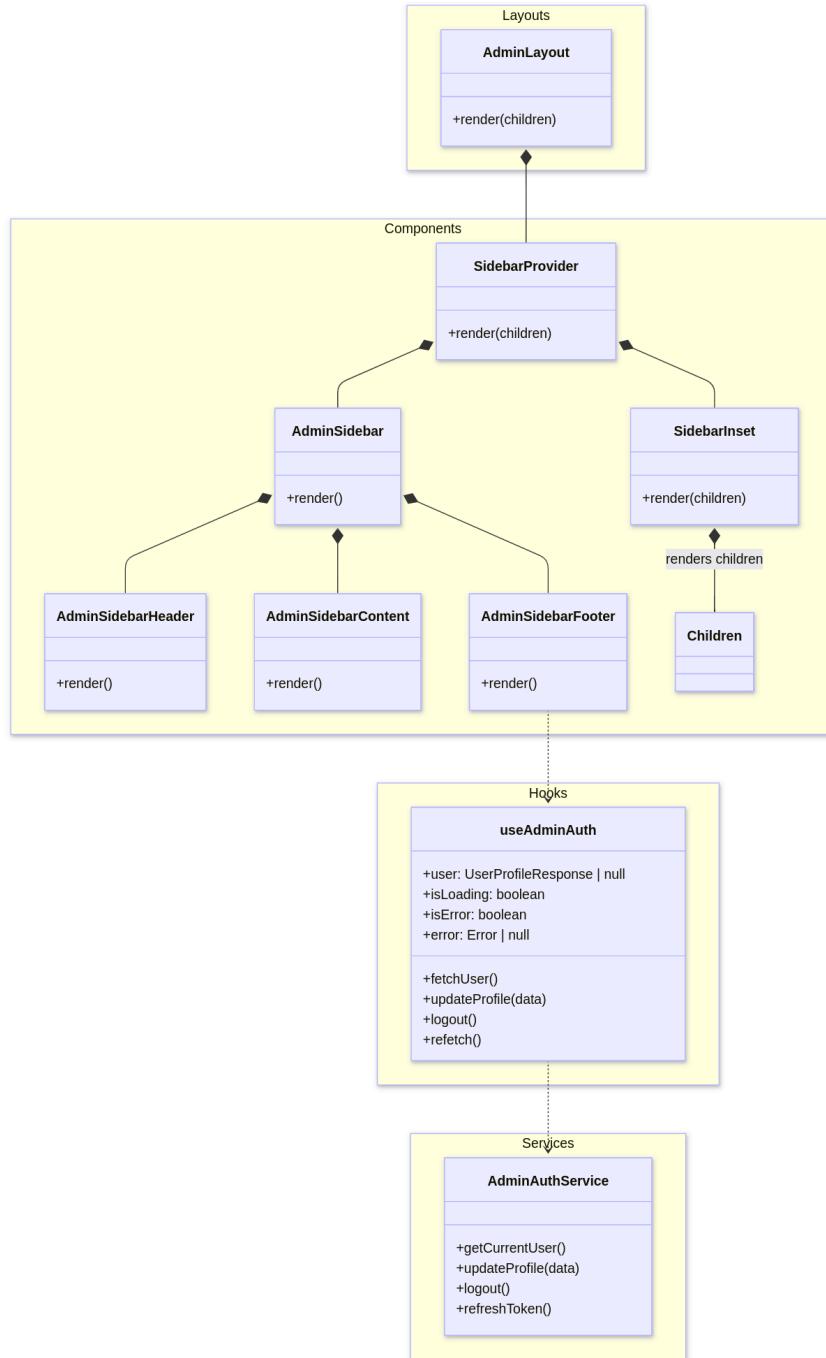
Services

- **CartService:** Handles API calls related to cart operations and synchronizes them with the server.
- **OrderService:** Handles API calls for order creation and payment processing.

Admin Pages

1. Layout

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Components

- **AdminLayout:** The top-level wrapper component that structures the application page, managing the sidebar and main content areas.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **SidebarProvider**: A context provider that manages the state of the sidebar (collapsed/expanded) and provides this context to child components.
- **SidebarInset**: The wrapper for the main content area that adjusts its layout based on the sidebar's state.
- **AdminSidebar**: The main container for the navigation sidebar element.
- **AdminSidebarHeader**: The top section of the sidebar, typically containing the logo or branding.
- **AdminSidebarContent**: The middle section of the sidebar that renders the scrollable list of navigation menu items.
- **AdminSidebarFooter**: The bottom section of the sidebar, used for displaying the current user's profile and logout controls.

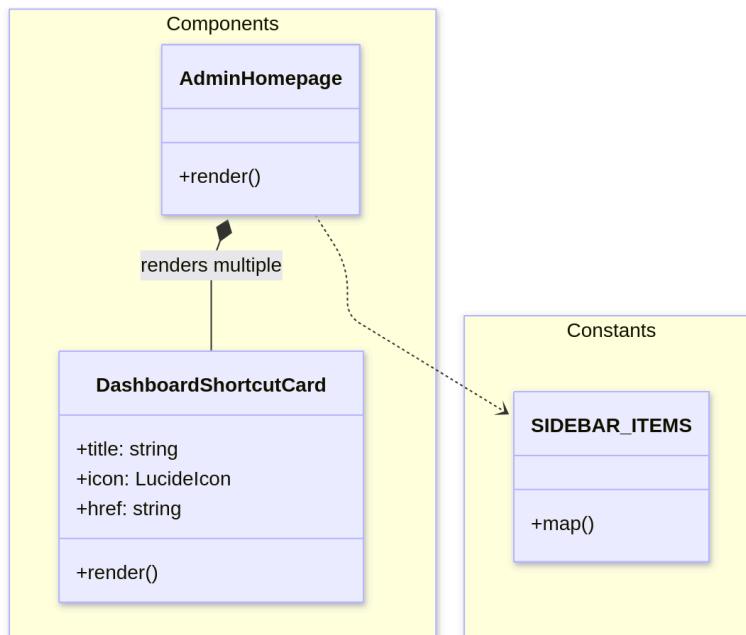
Hooks

- **useAdminAuth**: A custom hook that manages the administrator's authentication state, including fetching the current user profile, updating profile details, and handling logout logic.

Services

- **AdminAuthService**: The API service responsible for communicating with backend authentication endpoints (login, profile updates, token refresh).

2. Homepage



Components

- **AdminHomepage**: The main view component for the landing page, responsible for rendering the overall layout of shortcuts.

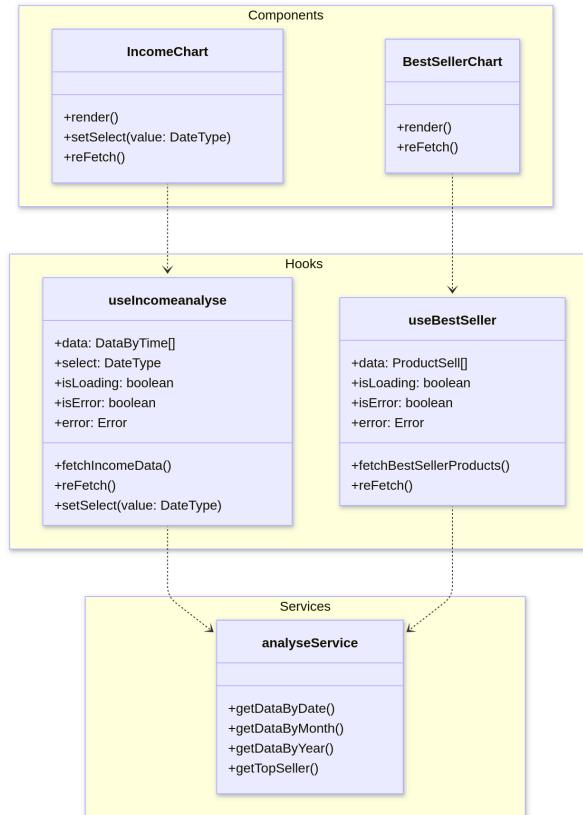
AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **DashboardShortcutCard**: A reusable UI component that displays a quick-access card (with title, icon, and link) to a specific admin section.

Constants

- **SIDE BAR _ITEMS**: A constant definition containing the configuration for navigation items, which is mapped to generate the dashboard shortcuts and sidebar menu.

3. Dashboard



Components

- **BestSellerChart**: A visualization component (e.g., Bar or Pie chart) that renders the top-selling products data.
- **IncomeChart**: A visualization component (e.g., Line or Area chart) that displays revenue trends over selected time periods.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

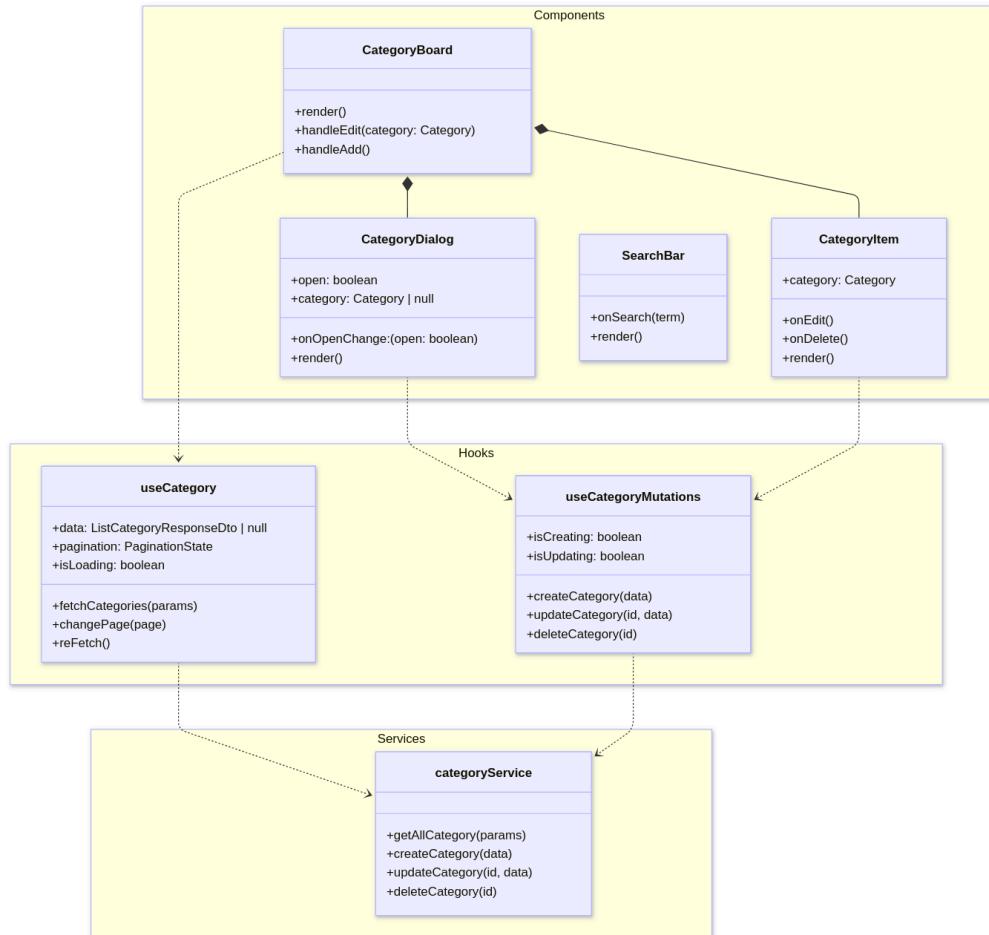
Hooks

- **useIncomeanalyse**: Manages the state and data fetching for revenue analysis, including handling time-period selections (Day/Month/Year).
- **useBestSeller**: Manages the state and data fetching for the best-seller reports.

Services

- **analyseService**: The API service responsible for fetching analytical data from the backend.

4. Category Management



Components

- **CategoryBoard**: The main container component for the category section, orchestrating the list view and action buttons.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **CategoryItem**: A presentation component representing a single category row or card with edit/delete actions.
- **CategoryDialog**: A modal component used for both creating new categories and editing existing ones.
- **SearchBar**: A reusable input component for filtering the category list by name.

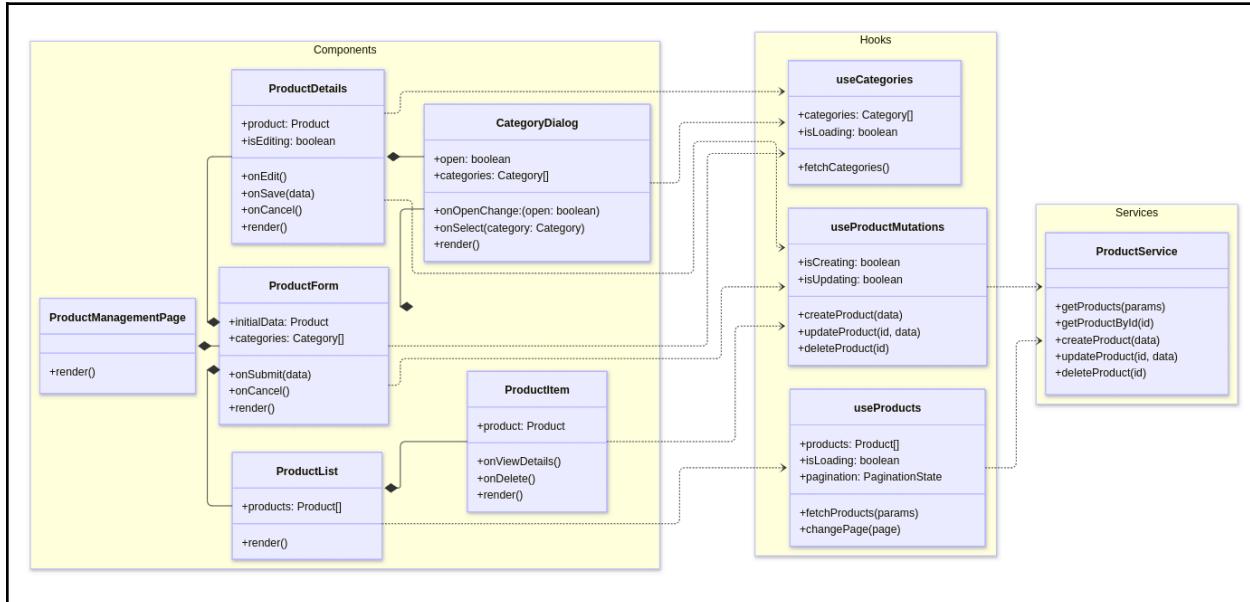
Hooks

- **useCategory**: Manages the reading state of categories, including fetching data, handling pagination, and loading states.
- **useCategoryMutations**: Manages the write operations (Create, Update, Delete) and their associated loading states.

Services

- **categoryService**: The API service responsible for CRUD operations on category entities.

5. Product Management



Components

- **ProductManagementPage**: The main page wrapper for the product management feature.
- **ProductList**: A component responsible for rendering the grid or table of products.
- **ProductItem**: A row or card component representing specific product details and actions.
- **ProductForm**: A form component handling input for creating new products, including category selection.
- **ProductDetails**: A component for viewing and editing detailed information of an existing product.
- **CategoryDialog**: A reused or specific dialog component allowing the selection of categories during product creation/editing.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

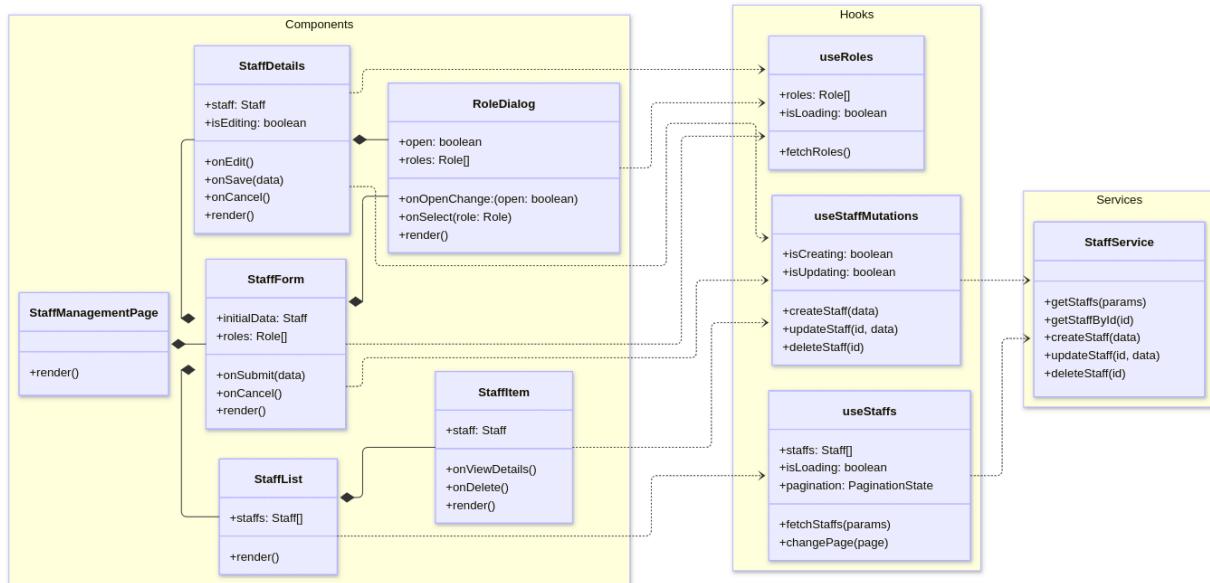
Hooks

- **useProducts**: Manages the state for fetching and paginating the product list.
- **useProductMutations**: Manages the state for Create, Update, and Delete operations on products.
- **useCategories**: A helper hook used within product forms to fetch the list of available categories for selection.

Services

- **ProductService**: The API service responsible for CRUD operations on product entities.

6. Staff Management



Components

- **StaffManagementPage**: The main page wrapper for the staff management feature.
- **StaffList**: A component responsible for rendering the list of staff members.
- **StaffItem**: A component representing a single staff member row with management actions.
- **StaffForm**: A form component used to register new staff members and assign initial roles.
- **StaffDetails**: A component for viewing and editing details of an existing staff member.
- **RoleDialog**: A dialog component used to search and select security roles to assign to a staff member.

Hooks

- **useStaffs**: Manages the fetching and pagination of staff member data.
- **useStaffMutations**: Manages the Create, Update, and Delete operations for staff accounts.

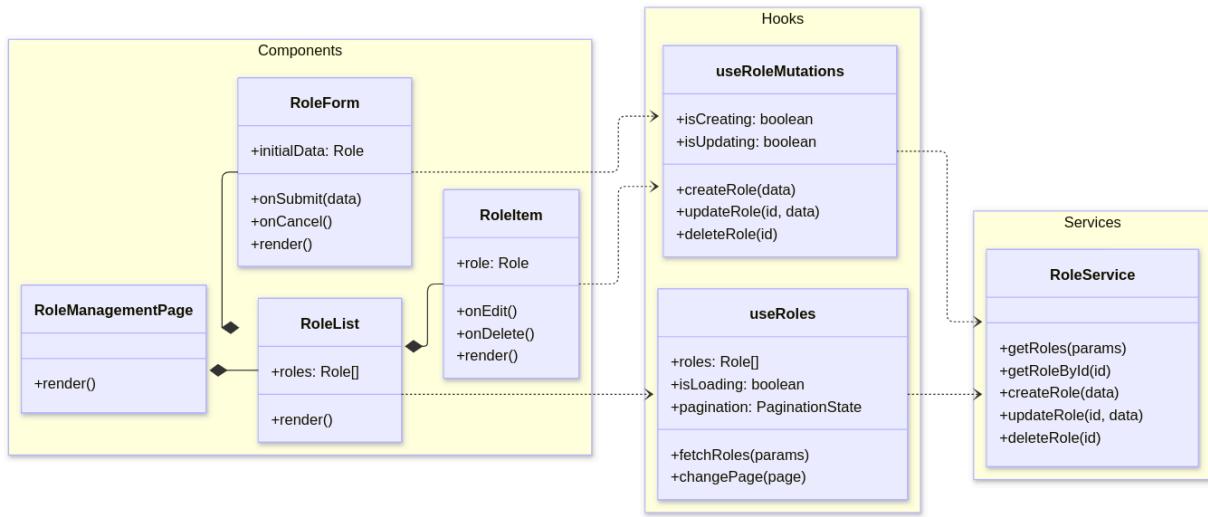
AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **useRoles**: A helper hook used to fetch available roles for assignment during staff creation or editing.

Services

- **StaffService**: The API service responsible for CRUD operations on staff entities.

7. Role Management



Components

- **RoleManagementPage**: The main page wrapper for the role management feature.
- **RoleList**: A component responsible for listing all defined system roles.
- **RoleItem**: A component representing a single role row with edit/delete actions.
- **RoleForm**: A form component used to define new roles or edit existing permissions.

Hooks

- **useRoles**: Manages the fetching and pagination of the roles list.
- **useRoleMutations**: Manages the Create, Update, and Delete operations for role definitions.

Services

- **RoleService**: The API service responsible for CRUD operations on role entities.

4.2 Component: Controllers

4.2.1 Responsibilities

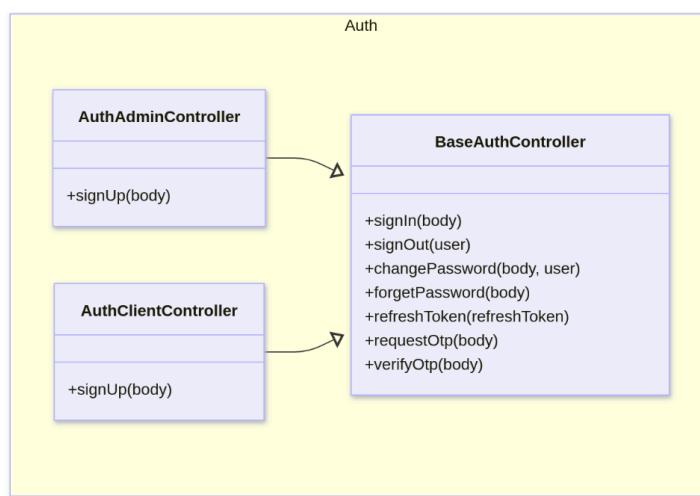
The Controller component acts as the **Presentation Layer** for the backend API. Its primary responsibilities are:

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

- **Request Handling:** Receiving HTTP requests (GET, POST, PUT, DELETE) from the client.
- **Orchestration:** Calling the appropriate Service layer methods to execute business logic.
- **Security Integration:** working with Middleware to ensure endpoints are protected by authentication and authorization rules.

4.2.2 Details

1. Auth



Base Auth Controller (Shared Logic)

Method Name	Functionality
signIn(body)	Authenticates a user via email/password and returns an Access/Refresh token pair.
signOut(user)	Invalidates the current user session or revoke tokens to log the user out.
refreshToken(token)	Issues a new Access Token using a valid Refresh Token to extend the session.
changePassword(body)	Updates the authenticated user's password.
forgetPassword(body)	Initiates the account recovery process.
requestOtp(body)	Generates and sends a One-Time Password.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

verifyOtp(body)	Validates a user-submitted OTP code.
------------------------	--------------------------------------

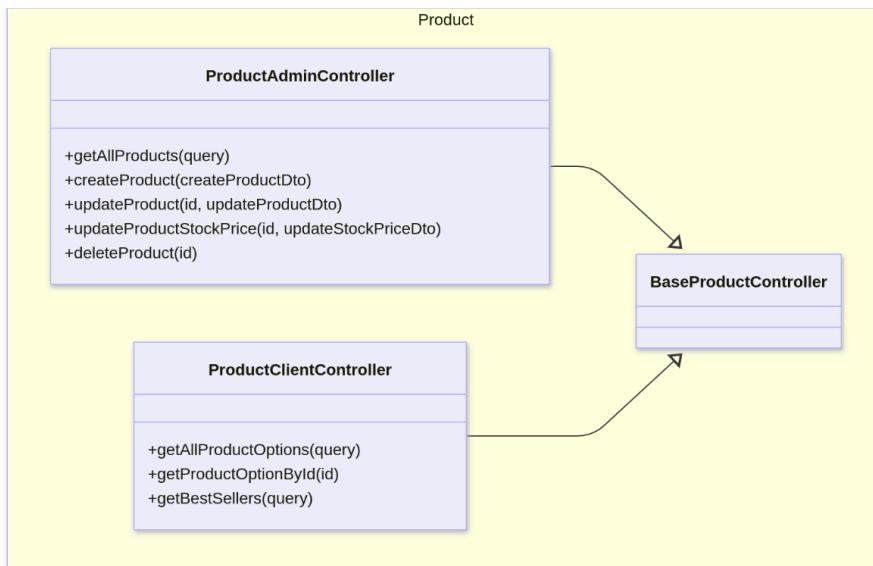
Auth Admin Controller

Method Name	Functionality
signUp(body)	Registers a new internal Admin/Staff account. Root admin will implement this action and assign new staff with appropriate role.

Auth Client Controller

Method Name	Functionality
signUp(body)	Registers a new Customer account (public registration flow).

2. Product



Base Product Controller

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Method Name	Functionality
<i>(None exposed directly, logic shared)</i>	Shared validation or helper logic.

Product Admin Controller

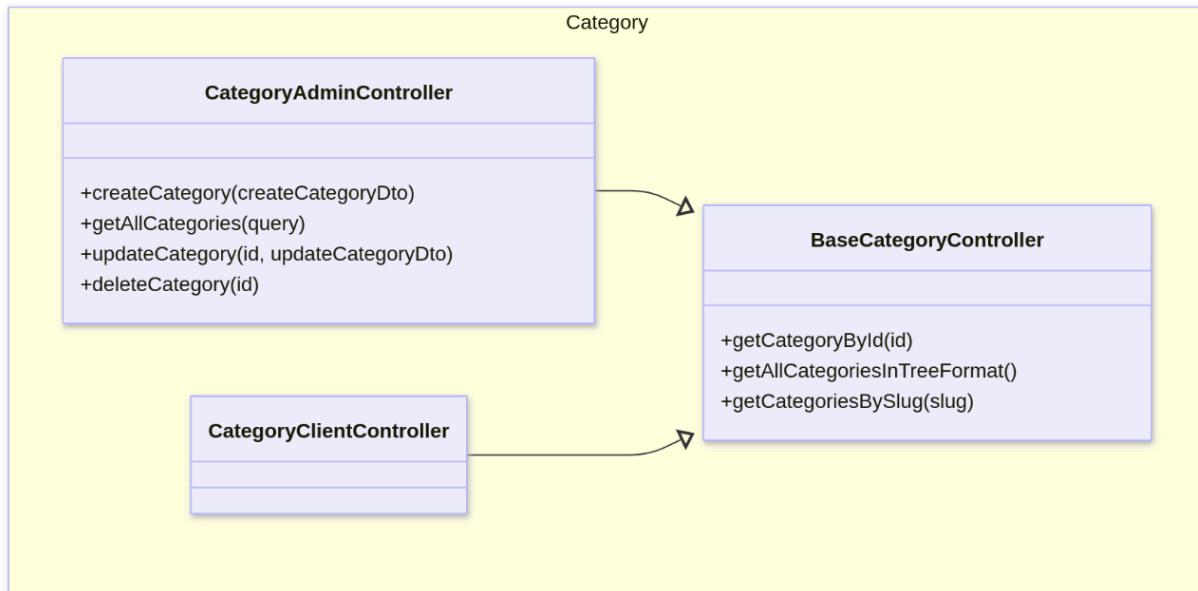
Method Name	Functionality
getAllProducts(query)	Retrieves a paginated list of all products (including inactive).
createProduct(dto)	Creates a new product entry along with its initial variants and options.
updateProduct(id, dto)	Updates general product metadata.
updateProductStockPrice(id, dto)	Updates stock quantity and price for a variant.
deleteProduct(id)	Soft-deletes or removes a product from the catalog.

Product Client Controller

Method Name	Functionality
getBestSellers(query)	Retrieves a list of top-selling products.
getAllProductOptions(query)	Retrieves available product attributes (e.g., all Colors/Sizes) for filtering.
getProductOptionById(id)	Retrieves details for a specific variant option.

3. Category

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Base Category Controller

Method Name	Functionality
getCategoryById(id)	Retrieves details of a specific category.
getAllCategoriesInTreeFormat()	Returns the full category structure nested as a tree.
getCategoriesBySlug(slug)	Finds a category using its URL-friendly slug text.

Category Admin Controller

Method Name	Functionality
getAllCategories(query)	Retrieves a flat, paginated list of categories for administration.
createCategory(dto)	Creates a new category node.

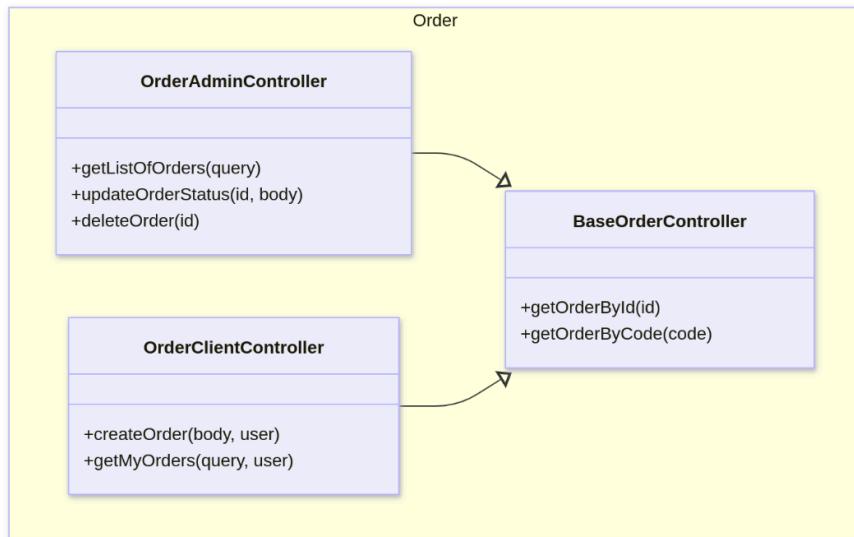
AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

updateCategory(id, dto)	Updates category details or moves it to a new parent.
deleteCategory(id)	Removes a category.

Category Client Controller

Method Name	Functionality
<i>(Inherits Base Read Operations)</i>	Used for public navigation menus.

4. Order



Base Order Controller

Method Name	Functionality
getOrderById(id)	Retrieves full details of a specific order.
getOrderByCode(code)	Retrieves an order using its public reference code.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

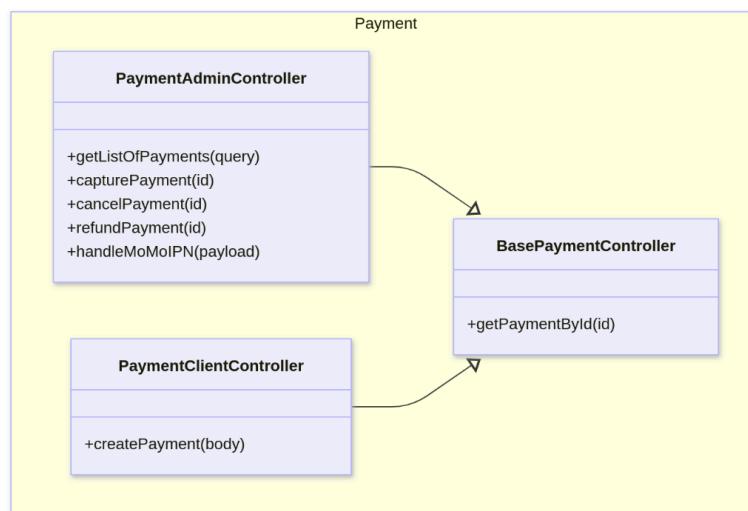
Order Admin Controller

Method Name	Functionality
getListOfOrders(query)	Retrieves a paginated, filterable list of all system orders.
updateOrderStatus(id, body)	Transitions an order to a new state (e.g., "Shipped").
deleteOrder(id)	Cancels or soft-deletes an order.

Order Client Controller

Method Name	Functionality
createOrder(body, user)	Converts a user's cart or selection into a persisted order record.
getMyOrders(query, user)	Retrieves the history of orders placed by the logged-in user.

5. Payment



Base Payment Controller

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Method Name	Functionality
getPaymentById(id)	Retrieves detailed logs of a specific payment transaction.

Payment Admin Controller

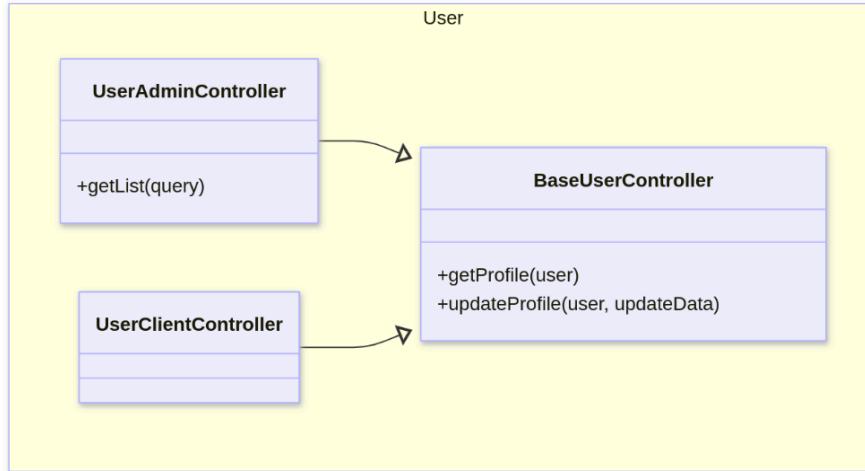
Method Name	Functionality
getListOfPayments(query)	Retrieves a history of all payment transactions for reconciliation.
capturePayment(id)	Manually captures an authorized payment.
cancelPayment(id)	Voids a pending payment transaction.
refundPayment(id)	Issues a refund for a previously successful transaction.
handleMoMoIPN(payload)	Webhook endpoint for MoMo gateway updates.

Payment Client Controller

Method Name	Functionality
createPayment(body)	Initializes a payment intent and returns checkout information.

6. User

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Base User Controller

Method Name	Functionality
getProfile(user)	Retrieves the profile information of the currently authenticated user.
updateProfile(user, data)	Updates personal details for the current user.

User Admin Controller

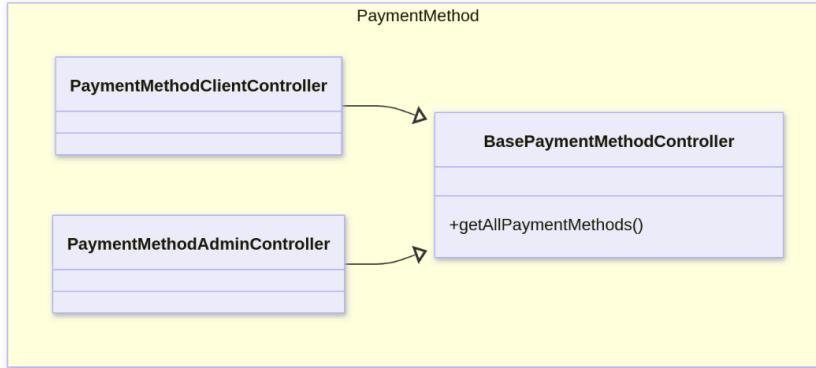
Method Name	Functionality
getList(query)	Retrieves a paginated list of all users in the system.

User Client Controller

Method Name	Functionality
<i>(Inherits Base Profile Operations)</i>	Standard user profile management.

7. Payment Method

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Base Payment Method Controller

Method Name	Functionality
getAllPaymentMethods()	Returns a list of active payment methods and their configuration.

Payment Method Admin Controller

Method Name	Functionality
<i>(Inherits Base Read Operations)</i>	Used during checkout to show options.

Payment Method Client Controller

Method Name	Functionality
<i>(Inherits Base Read Operations)</i>	Used during checkout to show options.

8. Role

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

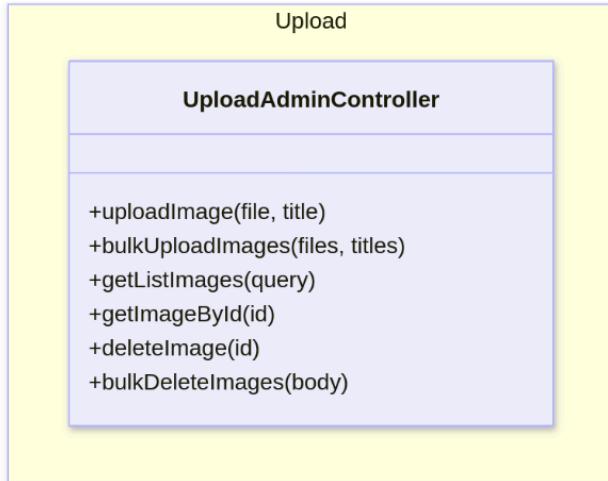


Role Admin Controller

Method Name	Functionality
<code>getListRoles(query)</code>	Retrieves all defined roles in the system.
<code>getRoleById(id)</code>	Retrieves a specific role and its list of permissions.
<code>createRole(body)</code>	Defines a new role with a specific set of permissions.
<code>updateRole(id, body)</code>	Modifies the name or permissions of an existing role.
<code>deleteRole(id)</code>	Removes a role from the system.

9. Upload

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

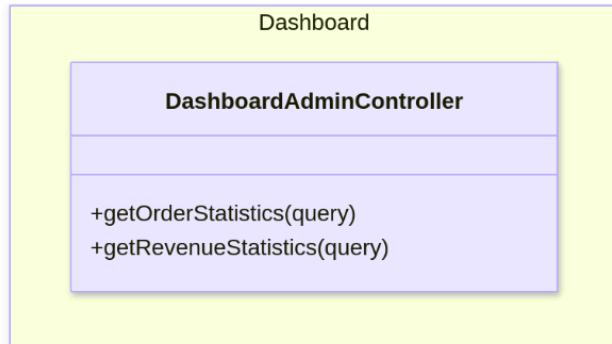


Upload Admin Controller

Method Name	Functionality
uploadImage(file, title)	Uploads a single file and returns its stored URL.
bulkUploadImages(files)	Handles the simultaneous upload of multiple files.
getListImages(query)	Retrieves a gallery of uploaded images.
getImageById(id)	Retrieves metadata for a specific uploaded image.
deleteImage(id)	Removes a single image file from storage.
bulkDeleteImages(body)	Removes multiple image files in a batch operation.

10. Dashboard

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Dashboard Admin Controller

Method Name	Functionality
<code>getOrderStatistics(query)</code>	Returns statistical data on orders (counts, status breakdown).
<code>getRevenueStatistics(query)</code>	Returns financial performance data over a specified period.

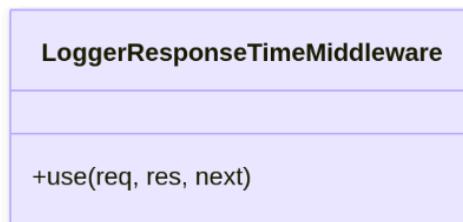
4.3 Component: Middleware

4.3.1 Responsibilities

These components handle cross-cutting concerns in the request/response lifecycle. They operate within the NestJS request pipeline to enforce security, validate data, handle errors, and transform responses.

4.3.2 Details

1. LoggerResponseTimeMiddleware

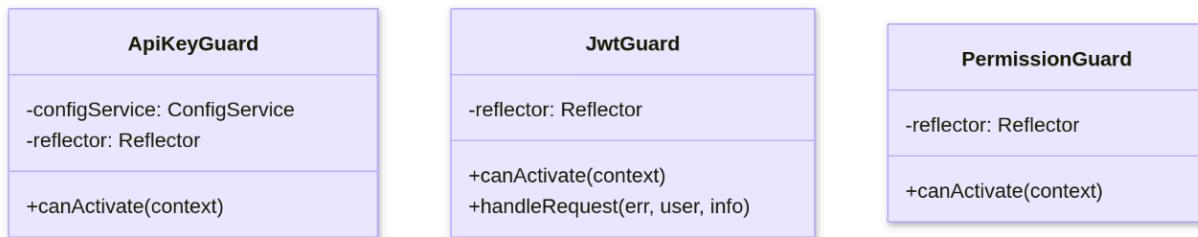


AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

This is a standard HTTP middleware that sits at the very entry point of the request pipeline. It helps monitoring the status of requests.

Method Name	Functionality
<code>use(req, res, next)</code>	Intercepts every incoming HTTP request to log the start time. It then hooks into the response 'finish' event to calculate and log the total duration (latency) of the request for monitoring performance.

2. Guard

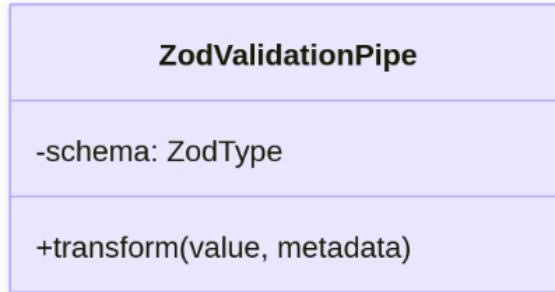


This is a global or route-specific Guard used to secure public-facing APIs that require machine-to-machine authentication (e.g., webhooks or mobile app initial headers).

Method Name	Functionality
ApiKeyGuard	Checks the incoming request headers for a specific API Key. It compares this key against the value stored in ConfigService. If they match, the request is allowed; otherwise, it is denied with a 403 Forbidden error.
JwtGuard	Validates the JWT found in the Authorization header. It verifies the signature and expiration. It also uses Reflector to check if the route is marked as "Public" (skipping validation if so).
PermissionGuard	Retrieves the required permissions defined for the route handler (using Reflector). It then checks if the currently authenticated user possesses these permissions. If not, it denies access.

5. Pipe

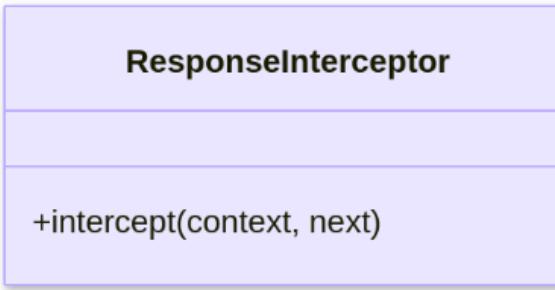
AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



This Pipe is responsible for request data validation. It ensures that the incoming request body, query params, or params match the expected schema structure defined using the Zod library.

Method Name	Functionality
<code>transform(value, metadata)</code>	Parses the incoming payload (value) against the provided Zod schema (schema). If validation fails, it throws a detailed BadRequestException; otherwise, it returns the sanitized/parsed data.

6. ResponseInterceptor



This Interceptor wraps the successful response from any controller handler into a standardized JSON envelope.

Method Name	Functionality

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

intercept(context, next)	Intercepts the response stream. It formats the data returned by the controller into a standard structure (e.g., { "data": ..., "statusCode": 200, "message": "Success" }) ensuring consistent API responses across the system.
---------------------------------	--

7. GlobalExceptionFilter



This Filter acts as the final safety net for the application. It catches any unhandled exceptions thrown during request processing.

Method Name	Functionality
catch(exception, host)	Catches errors (both HTTP and system errors). It standardizes the error response format (e.g., { "statusCode": 500, "error": "Internal Server Error", "timestamp": ... }) and logs the error details for debugging before sending the response to the client.

4.4 Component: Business Service

4.4.1 Responsibilities

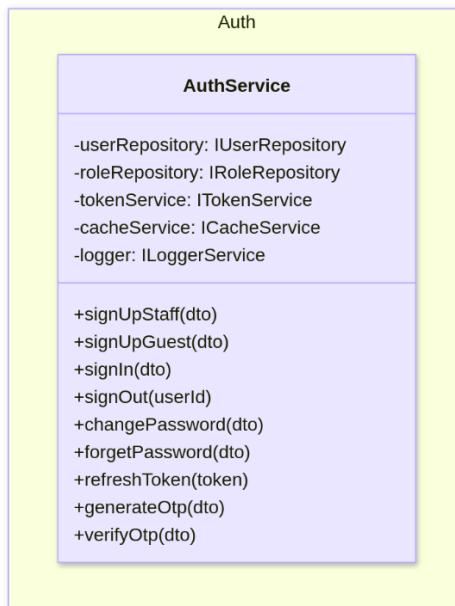
The Service component acts as the **Application Layer** (and sometimes Domain Service layer). It provides Data transfer objects (DTOs) for other components. Each service has its own class entities which represent the main data format. Its primary responsibilities are:

- **Business Logic Execution:** Implementing the core rules and calculations of the application.
- **Transaction Management:** Ensuring data consistency across multiple database operations via Unit of Work.
- **Orchestration:** Coordinating interactions between the Controller, Repositories, and External Providers.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

4.4.2 Details

1. Auth Service



Dependencies (Fields)

Field Name	Type	Description
userRepository	IUserRepository	Access to user persistence for credential validation.
roleRepository	IRoleRepository	Access to role definitions for assignment during signup.
tokenService	ITokenService	Utility for generating and signing JWTs.
cacheService	ICacheService	Temporary storage for OTPs and session data.
logger	ILoggerService	System logging interface.

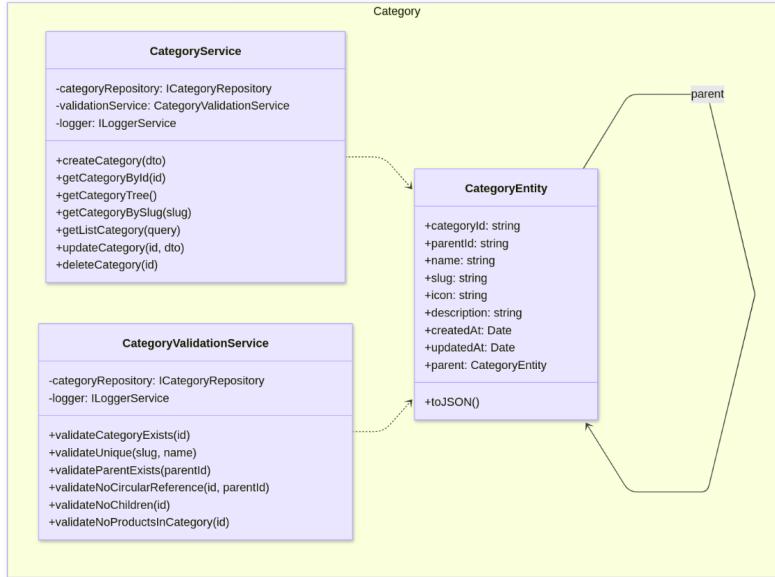
Methods

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Method Name	Functionality
signUpStaff(dto)	Registers a new internal user (admin/staff), ensuring appropriate role assignment and security checks.
signUpGuest(dto)	Registers a new external customer (guest), typically with default permissions.
signIn(dto)	Authenticates credentials against the stored hash and issues an access/refresh token pair.
signOut(userId)	Invalidates active sessions or tokens for the specified user.
changePassword(dto)	Verifies the old password and updates the user record with a new hashed password.
forgetPassword(dto)	Generates a recovery token or link and triggers the email notification service.
refreshToken(token)	Validates a long-lived refresh token and issues a new short-lived access token.
generateOtp(dto)	Creates a temporary numeric code for 2FA or verification and stores it with an expiration.
verifyOtp(dto)	Compares the user-submitted code against the active stored OTP.

2. Category Service

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Dependencies (Fields)

Field Name	Type	Description
categoryRepository	ICategoryRepository	Data access for category entities.
validationService	CategoryValidationService	Internal service for structural integrity checks.
logger	ILoggerService	System logging interface.

Methods

Method Name	Functionality
createCategory(dto)	Validates and persists a new category node in the database.
getCategoryById(id)	Retrieves a single category entity by its ID.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

getCategoryTree()	Recursively builds the full hierarchical structure of categories (Parent -> Children).
getCategoryBySlug(slug)	Finds a category based on its URL-friendly slug for SEO purposes.
getListCategory(query)	Retrieves a flat, paginated list of categories.
updateCategory(id, dto)	Modifies category details after performing necessary validation checks.
deleteCategory(id)	Removes a category (typically requires checking for child products first).

3. Category Validation Service

Dependencies (Fields)

Field Name	Type	Description
categoryRepository	ICategoryRepository	Access to check existence and relationships.
logger	ILoggerService	System logging interface.

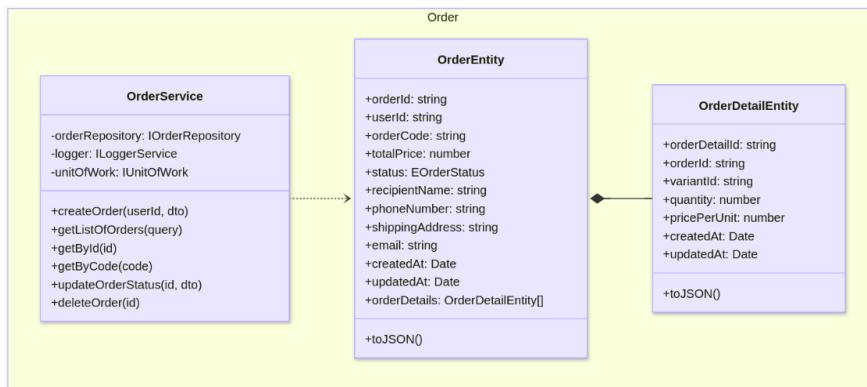
Methods

Method Name	Functionality
validateCategoryExists(id)	Checks if a category ID corresponds to a valid record in the database.
validateUnique(slug, name)	Ensures no two categories share the same URL slug or display name.
validateParentExists(parentId)	Ensures a referenced parent category actually exists before linking.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

validateNoCircularReference(id, parentId)	Prevents a category from becoming its own ancestor in the hierarchy.
validateNoChildren(id)	Checks if a category has sub-categories before allowing deletion.
validateNoProductsInCategory(id)	Checks if a category contains active products before allowing deletion.

4. Order Service



Dependencies (Fields)

Field Name	Type	Description
orderRepository	IOrderRepository	Data access for orders and line items.
unitOfWork	IUnitOfWork	Manages transactions for multi-step updates (e.g., stock + order).
logger	ILoggerService	System logging interface.

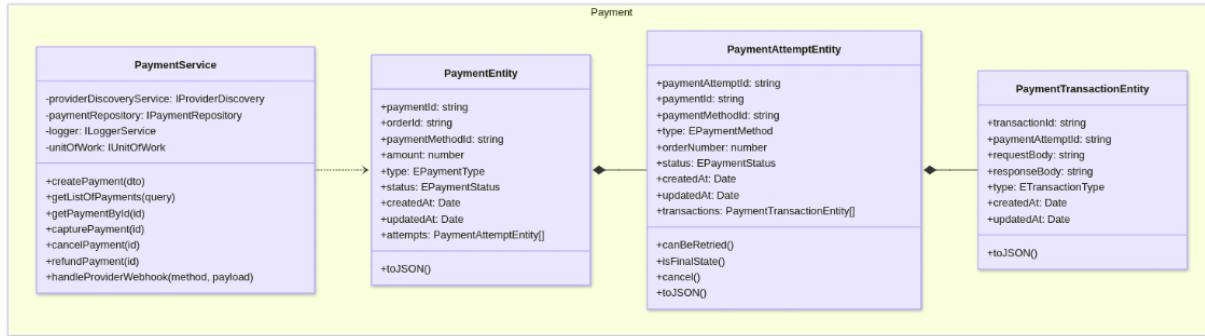
Methods

Method Name	Functionality

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

createOrder (userId, dto)	Converts cart items into a persisted order, calculates final prices, and reserves stock.
getListOfOrders (query)	Retrieves paginated orders for admin views with filtering capabilities.
getById (id)	Retrieves the full order aggregate including all line items (OrderDetail).
getByCode (code)	Retrieves an order by its public-facing reference number.
updateOrderStatus (id, dto)	Transitions the order state (e.g., Pending -> Shipped) and triggers related events.
deleteOrder (id)	Soft-deletes or cancels an order record.

5. Payment Service



Dependencies (Fields)

Field Name	Type	Description
providerDiscoveryService	IProviderDiscovery	Resolves the correct gateway strategy (MoMo, Stripe) at runtime.
paymentRepository	IPaymentRepository	Data access for payment records.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

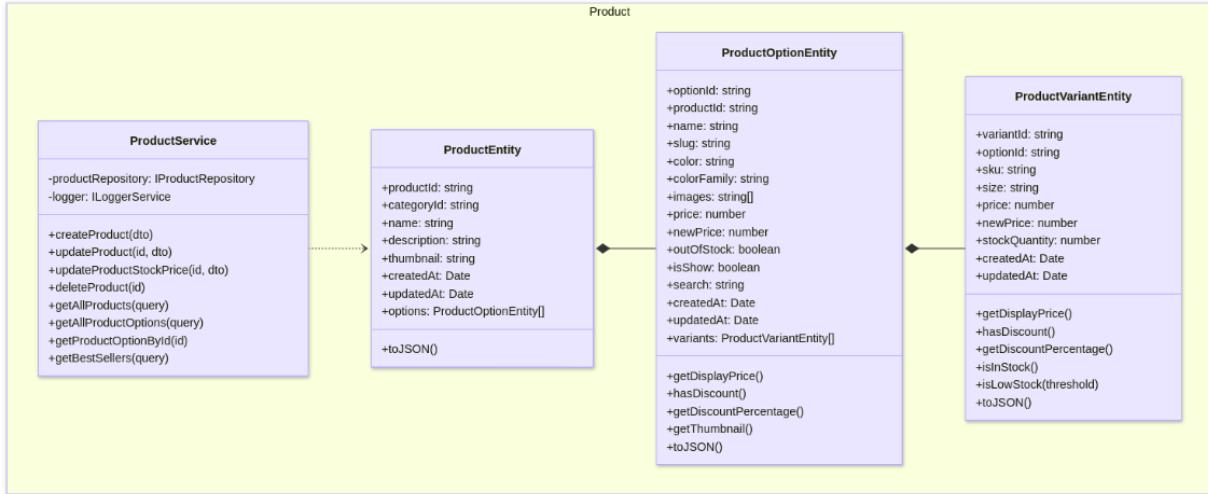
unitOfWork	IUnitOfWork	Ensures atomicity when updating payment status and order status.
logger	ILoggerService	System logging interface.

Methods

Method Name	Functionality
createPayment(dto)	Initializes a payment intent and returns checkout data/URLs.
getListOfPayments(query)	Retrieves payment history for reconciliation and auditing.
getPaymentById(id)	Retrieves full details of a payment transaction and its attempts.
capturePayment(id)	Finalizes an authorized payment (for two-step auth-capture flows).
cancelPayment(id)	Voids a pending payment attempt.
refundPayment(id)	Processes a refund for a successful payment.
handleProviderWebhook(method, payload)	Processes asynchronous status updates from gateways (MoMo).

6. Product Service

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	



Dependencies (Fields)

Field Name	Type	Description
productRepository	IProductRepository	Data access for the product aggregate.
logger	ILoggerService	System logging interface.

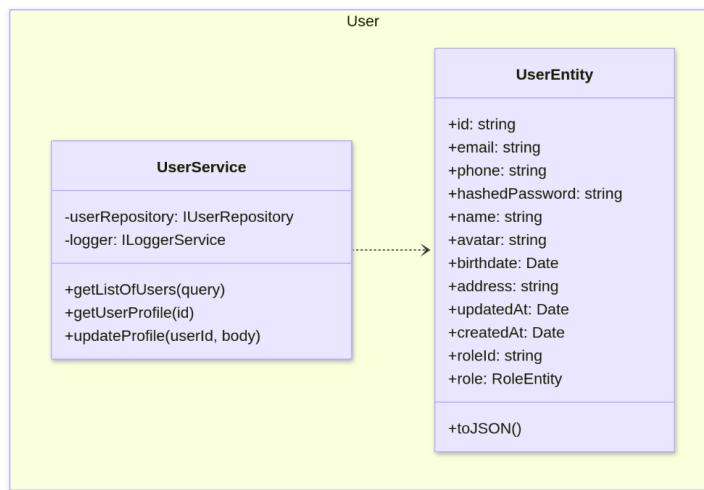
Methods

Method Name	Functionality
createProduct(dto)	Creates a product and its nested options/variants in a single transaction.
updateProduct(id, dto)	Updates general product metadata (name, description).
updateProductStockPrice(id, dto)	Efficiently updates inventory and pricing for specific variants.
deleteProduct(id)	Removes a product and all associated variants from the catalog.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

getAllProducts(query)	Retrieves the full admin list of products.
getAllProductOptions(query)	Retrieves available filter attributes (e.g., all Colors).
getProductOptionById(id)	Retrieves a specific option configuration.
getBestSellers(query)	Calculates top products based on historical sales data.

7. User Service



Dependencies (Fields)

Field Name	Type	Description
userRepository	IUserRepository	Access to user profile data.
logger	ILoggerService	System logging interface.

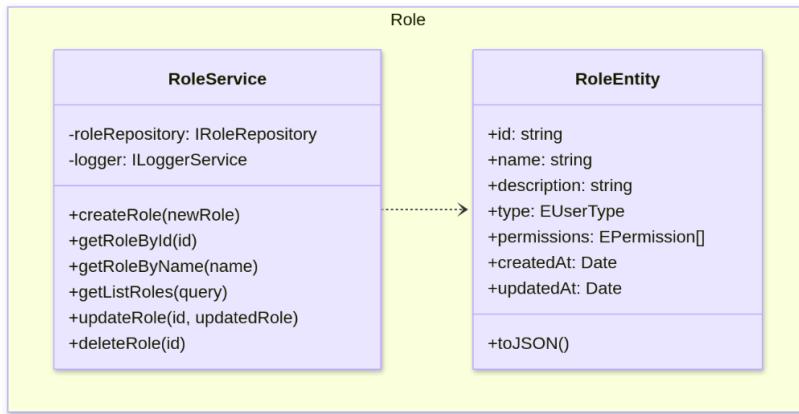
Methods

Method Name	Functionality

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

getListOfUsers(query)	Lists registered accounts for administration.
getUserProfile(id)	Retrieves sensitive profile data for the account owner.
updateProfile(userId, body)	Updates personal info like address, phone number, or avatar.

8. Role Service



Dependencies (Fields)

Field Name	Type	Description
roleRepository	IRoleRepository	Access to role and permission storage.
logger	ILoggerService	System logging interface.

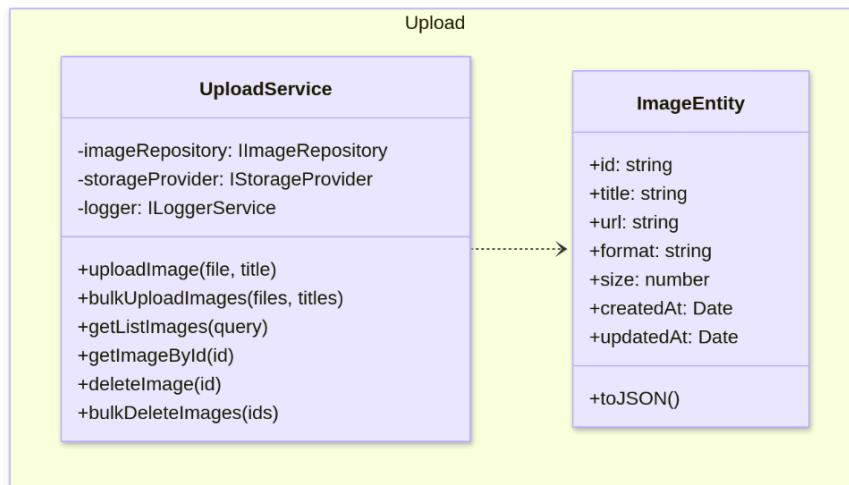
Methods

Method Name	Functionality
createRole(newRole)	Defines a new security role with specific permissions.
getRoleById(id)	Retrieves a role definition.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

getRoleByName(name)	Lookup utility for internal logic (e.g., finding the "Admin" role).
getListRoles(query)	Lists all available system roles.
updateRole(id, updatedRole)	Modifies permissions attached to a specific role.
deleteRole(id)	Removes a role (typically checks for usage first).

9. Upload Service



Dependencies (Fields)

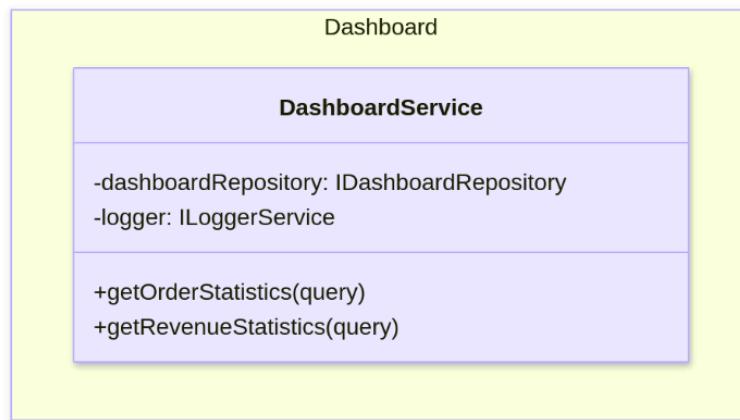
Field Name	Type	Description
imageRepository	IImageRepository	Stores metadata about the file (URL, size).
storageProvider	IStorageProvider	Handles the physical file I/O (S3, Disk).
logger	ILoggerService	System logging interface.

Methods

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Method Name	Functionality
uploadImage(file, title)	Processes a file upload stream and saves metadata.
bulkUploadImages(files, titles)	Handles batched file uploads.
getListImages(query)	Retrieves a gallery of uploaded assets.
getImageById(id)	Retrieves metadata for a specific file.
deleteImage(id)	Removes the file from storage and the database.
bulkDeleteImages(ids)	Batch deletion of multiple files.

10. Dashboard Service



Dependencies (Fields)

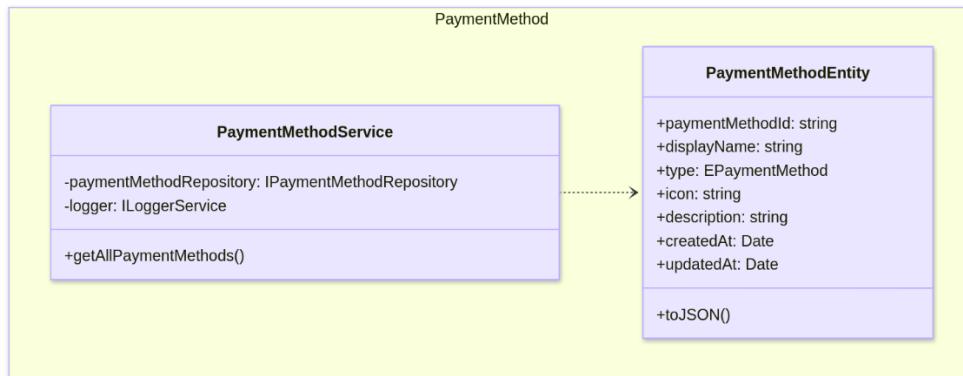
Field Name	Type	Description
dashboardRepository	IDashboardRepository	Specialized read-only repository for analytics queries.
logger	ILoggerService	System logging interface.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Methods

Method Name	Functionality
getOrderStatistics(query)	Aggregates order data by status/date.
getRevenueStatistics(query)	Calculates financial totals over time.

11. Payment Method Service



Dependencies (Fields)

Field Name	Type	Description
paymentMethodRepository	IPaymentMethodRepository	Access to configured payment methods.
logger	ILoggerService	System logging interface.

Methods

Method Name	Functionality
getAllPaymentMethods()	Lists enabled payment options for the checkout UI.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

4.5 Component: Repository

4.5.1 Responsibilities

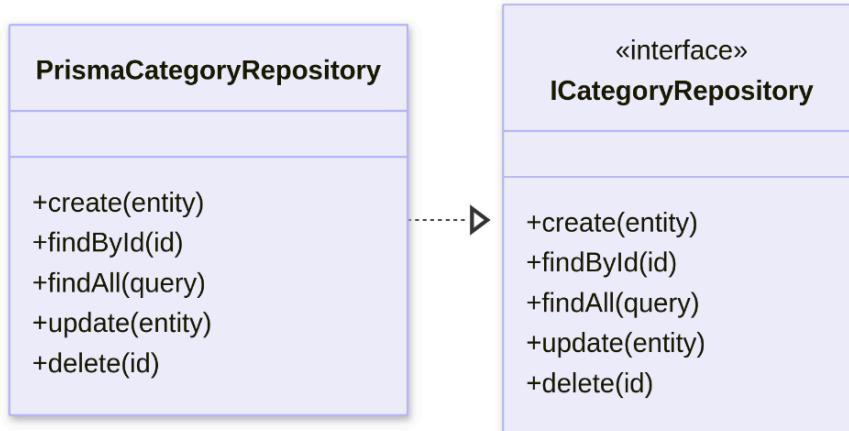
The Repository component acts as the Infrastructure Layer (specifically the Data Access layer). Each domain has specific repository interfaces that keep the service layer isolated from technology. We can easily change to a new database but not affect other components. Its primary responsibilities are:

- **Data Persistence:** Abstracting the underlying database technology (SQL, NoSQL, etc.) from the rest of the application.
- **Query Encapsulation:** containing complex database queries (e.g., joins, aggregations) so services remain clean.
- **Entity Mapping:** Converting raw database rows/documents into Domain Entities.

In this project, we choose Prisma to handle database connection, so each implementation of the repository has the prefix Prisma.

4.5.2 Details

1. Category Repository

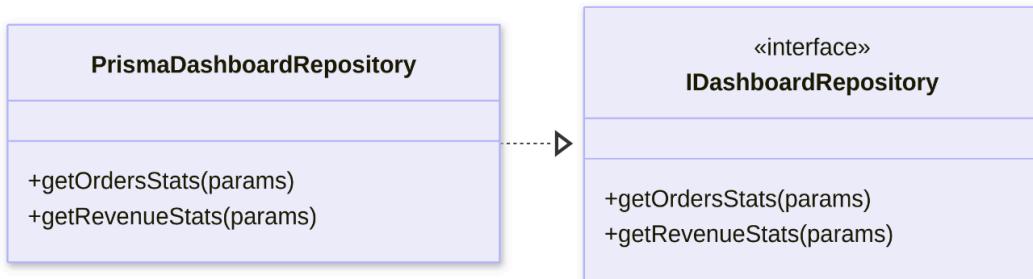


Method Name	Functionality
<code>create(entity)</code>	Persists a new category entity into the database.
<code>findById(id)</code>	Retrieves a specific category by its unique identifier.
<code>findAll(query)</code>	Retrieves a list of categories based on filtering criteria.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

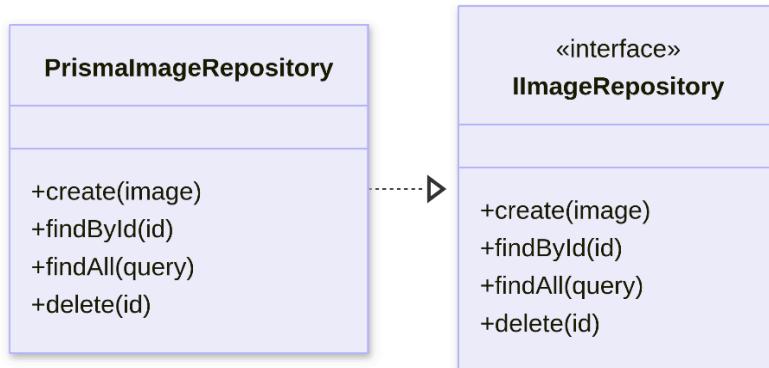
update(entity)	Updates the attributes of an existing category record.
delete(id)	Removes a category record from the database.

2. Dashboard Repository



Method Name	Functionality
getOrdersStats(params)	Aggregates order data (counts, status distribution) based on time range parameters.
getRevenueStats(params)	Calculates total revenue and financial metrics based on time range parameters.

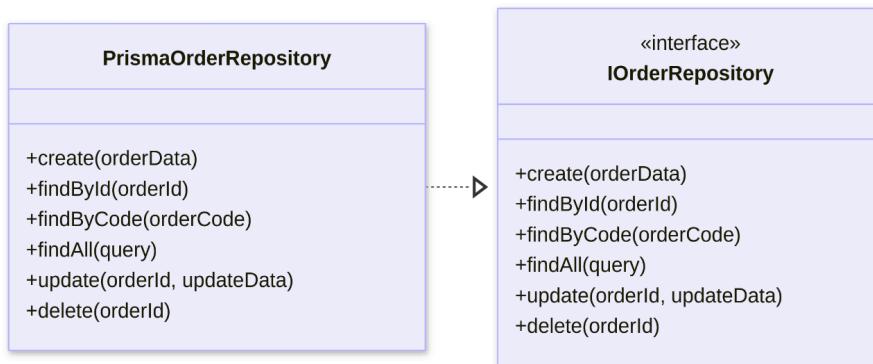
3. Image Repository



AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Method Name	Functionality
create(image)	Saves the metadata of a newly uploaded image.
findById(id)	Retrieves image metadata by its ID.
findAll(query)	Retrieves a gallery of image records based on filters.
delete(id)	Removes an image metadata record.

4. Order Repository

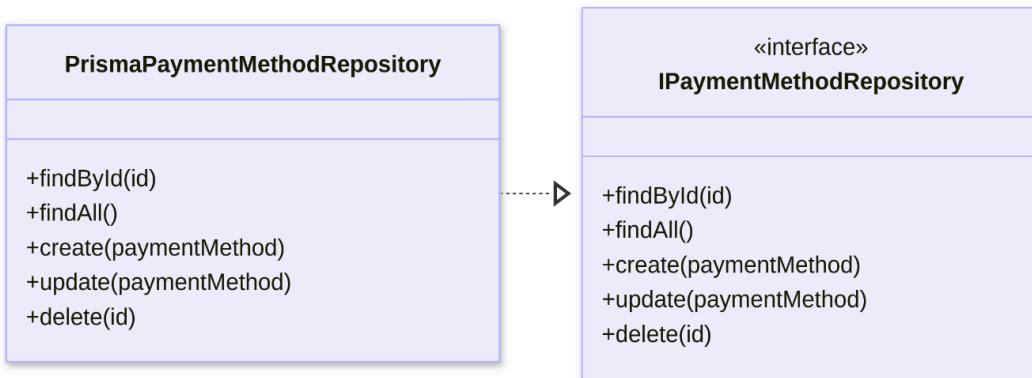


Method Name	Functionality
create(orderData)	Inserts a new order and its details into the database.
findById(orderId)	Retrieves a full order aggregate by its internal ID.
findByCode(orderCode)	Retrieves an order using its public-facing reference code.
findAll(query)	Retrieves a paginated list of orders for administration.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

update(orderId, updateData)	Modifies order status or details.
delete(orderId)	Soft-deletes or removes an order record.

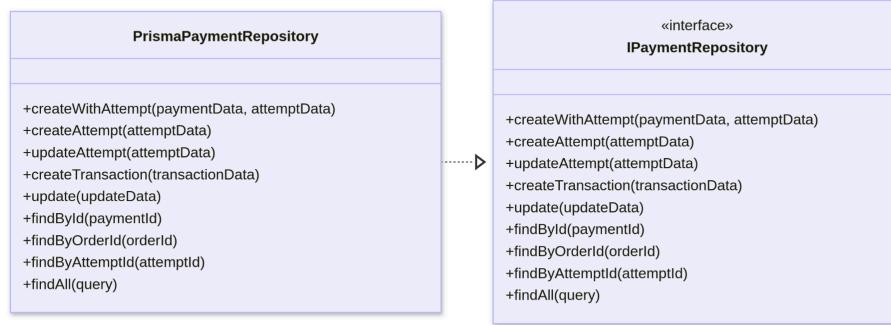
5. Payment Method Repository



Method Name	Functionality
findById(id)	Retrieves a specific payment method configuration.
findAll()	Retrieves all available payment methods.
create(paymentMethod)	Adds a new supported payment gateway configuration.
update(paymentMethod)	Modifies an existing payment method configuration.
delete(id)	Removes a payment method configuration.

6. Payment Repository

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

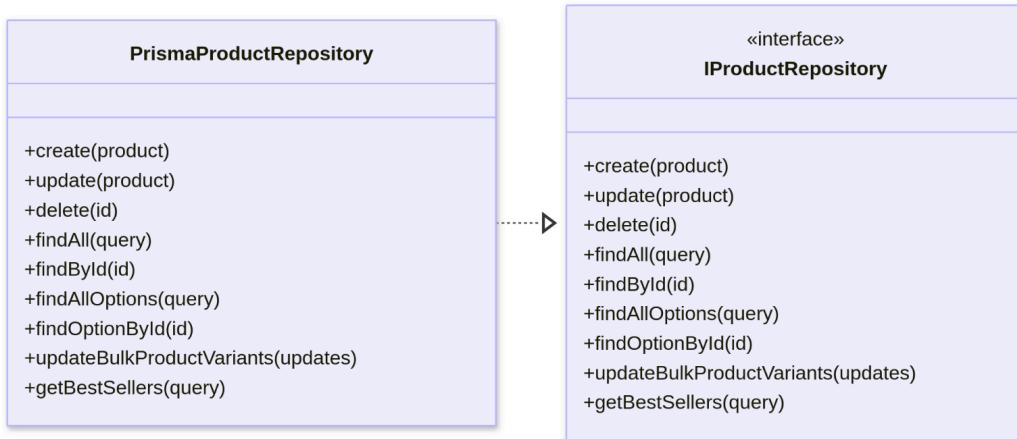


Method Name	Functionality
createWithAttempt (paymentData, attemptData)	Atomically creates a payment record and its first attempt.
createAttempt (attemptData)	Logs a new attempt for an existing payment obligation.
updateAttempt (attemptData)	Updates the status of a specific payment attempt.
createTransaction (transactionData)	Logs a raw provider transaction (request/response) for auditing.
update (updateData)	Updates the overall status of the parent payment record.
findById (paymentId)	Retrieves the full payment aggregate.
findByPrimaryKey (orderId)	Finds the payment record associated with a specific order.
findByAttemptId (attemptId)	Finds a payment record via one of its attempt IDs.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

findAll(query)	Retrieves a history of payments for reconciliation.
-----------------------	---

7. Product Repository

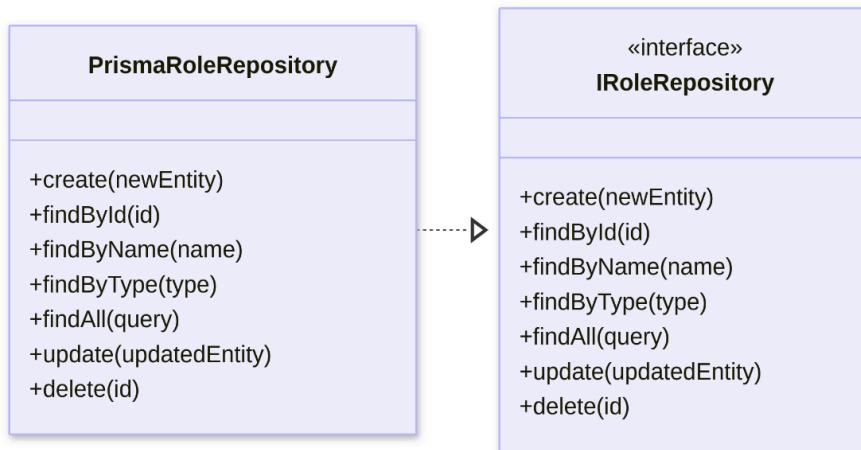


Method Name	Functionality
create(product)	Persists a new product aggregate (including options/variants).
update(product)	Updates product details.
delete(id)	Removes a product from the catalog.
findAll(query)	Retrieves a filterable list of products.
findById(id)	Retrieves a full product aggregate.
findAllOptions(query)	Retrieves distinct product options (e.g., all available colors).

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

findOptionById(id)	Retrieves a specific option configuration.
updateBulkProductVariants(updates)	Efficiently performs batch updates on inventory/pricing.
getBestSellers(query)	specific query to rank products by sales volume.

8. Role Repository

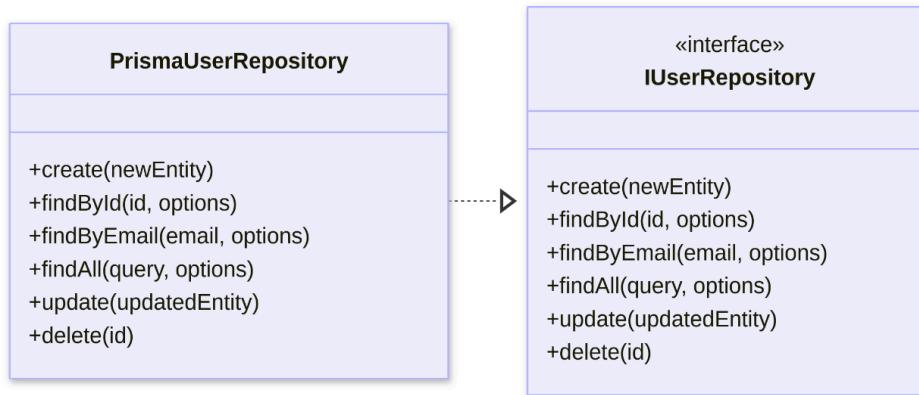


Method Name	Functionality
create(newEntity)	Persists a new role definition.
findById(id)	Retrieves a role and its permissions by ID.
findByName(name)	Retrieves a role by its unique name (e.g., "Admin").
findByType(type)	Finds roles associated with a specific user type.
findAll(query)	Retrieves all defined roles.
update(updatedEntity)	Modifies role permissions or metadata.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

delete(id)	Removes a role definition.
-------------------	----------------------------

9. User Repository



Method Name	Functionality
create(newEntity)	Persists a new user record.
findById(id, options)	Retrieves a user by ID, optionally including relations.
findByEmail(email, options)	Retrieves a user by their unique email address.
findAll(query, options)	Retrieves a paginated list of users.
update(updatedEntity)	Updates user profile or account details.
delete(id)	Removes a user record.

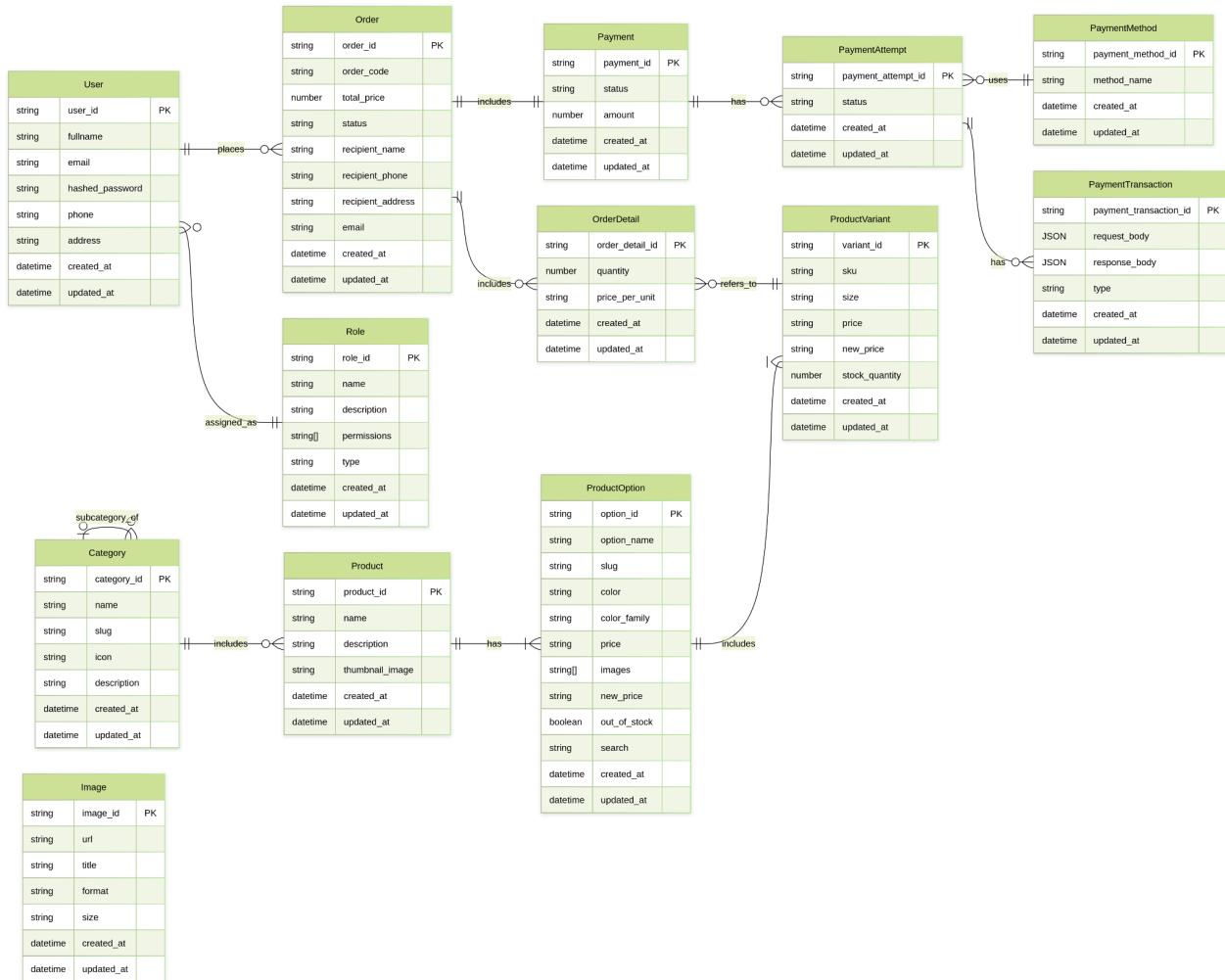
4.6 Component: Database

For the database, we have selected SQL database. This choice is appropriate because it guarantees strict data consistency, which is critical given the high frequency of orders and payments. Each entity identified

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

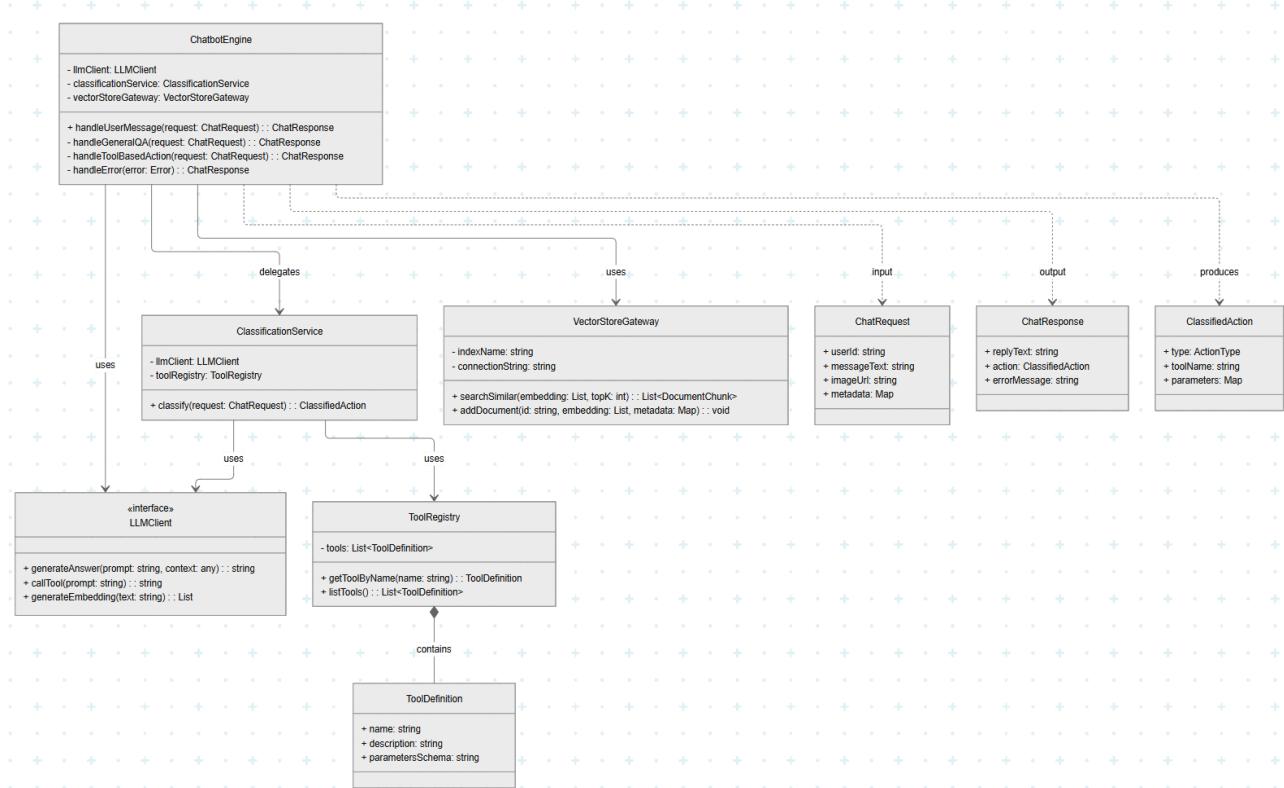
in the class diagram will correspond to a specific table within the database allowing the data model to remain clear, normalized, and easy to maintain.

Only the backend can communicate with the database. It will ensure more security and loose the couple between other components. The backend communicates with the database through a combination of the Infrastructure layer and the SQL driver provided by our cloud database service (Supabase). The infrastructure layer contains class repositories that implement the interfaces defined in the Core layer. Communication occurs over secure TCP/IP connections, ensuring that all database operations remain safe and protected.



AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

4.7 Component: AI



4.7.1. ChatbotEngine

a) Description

The ChatbotEngine is the core orchestrator of the AI chatbot system. It receives user messages, retrieves relevant context via RAG, invokes the LLM for reasoning, and delegates system-level actions to the classification layer. It is responsible for producing the final ChatResponse returned to the web application.

b) Fields

Field Name	Type	Description
llmClient	LLMClient	Provides LLM capabilities such as generation, tool-calling, and embedding.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

classificationService	ClassificationService	Determines which tool/function the LLM intends to execute.
vectorStoreGateway	VectorStoreGateway	Performs vector search for Retrieval-Augmented Generation (RAG).

c) Methods

Method Name	Functionality
handleUserMessage(request: ChatRequest)	Main entry point. Processes a user message and returns a final ChatResponse.
handleGeneralQA(request: ChatRequest)	Handles natural language Q&A using RAG + LLM reasoning.
handleToolBasedAction(request: ChatRequest)	Executes system actions chosen by the LLM (via ClassifiedAction).
handleError(error: Error)	Produces a standardized error response in case of failure.

4.7.2. LLMClient

a) Description

The LLMClient wraps the underlying language model API (groq). It generates responses, produces embeddings, and performs tool-calling for structured action selection.

b) Fields

Field Name	Type	Description
apiKey	string	API key for authenticating with the LLM provider.
modelName	string	The LLM model being used (e.g., "mixtral", "llama3").
endpointUrl	string	Base URL of the LLM service.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

c) Methods

Method Name	Functionality
generateAnswer(prompt: string, context: any)	Generates a natural language response using the model.
callTool(prompt: string)	Returns JSON representing the LLM's selected tool/function.
generateEmbedding(text: string)	Produces vector embeddings for similarity search (RAG).

4.7.3. ClassifiedAction

a) Description

ClassifiedAction is the structured output representing which system tool/function the LLM wants to execute and with what parameters. (*This class contains no methods, it is a pure data carrier.*)

b) Fields

Field Name	Type	Description
type	ActionType	The type/category of action requested (e.g., TRY_ON, GET_PRICE).
toolName	string	The name of the selected tool from the registry.
parameters	Map<string, any>	Dynamic parameters passed to the tool.

4.7.4. ClassificationService

a) Description

ClassificationService is responsible for interpreting the user input and LLM output to determine which tool should be executed. It acts as the bridge between free-text reasoning and actionable system commands.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

b) Fields

Field Name	Type	Description
llmClient	LLMClient	Used to perform tool selection via LLM.
toolRegistry	ToolRegistry	Contains all available tools and their metadata.

c) Methods

Method Name	Functionality
classify(request: ChatRequest)	Uses the LLM to identify the intended tool and produce a ClassifiedAction.

4.7.5. ToolRegistry

a) Description

ToolRegistry stores the list of available tools that the LLM may call. It enables dynamic tool lookup and ensures the chatbot has access only to approved system actions.

b) Fields

Field Name	Type	Description
tools	List<ToolDefinition>	The complete list of registered tools/functions.

c) Methods

Method Name	Functionality
getToolByName(name: string)	Retrieves a tool definition by name.
listTools()	Returns all available tool definitions.

4.7.6. ToolDefinition

a) Description

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Defines a single tool/function that the LLM can call. Each tool has a name, description, and parameter schema for validation. (This class contains no methods, it is metadata only.)

b) Fields

Field Name	Type	Description
name	string	Tool name as referenced by the LLM.
description	string	Human-readable description of the tool's functionality.
parametersSchema	string	JSON schema describing required parameters.

4.7.7. VectorStoreGateway

a) Description

The VectorStoreGateway enables Retrieval-Augmented Generation (RAG) by storing and retrieving document embeddings.

b) Fields

Field Name	Type	Description
indexName	string	Name of the vector index used for similarity search.
connectionString	string	Connection details for the vector database.

c) Methods

Method Name	Functionality
searchSimilar(embedding, topK)	Retrieves the top-K similar document chunks.
addDocument(id, embedding, metadata)	Inserts a new document into the vector store.

4.7.8 ChatRequest

a) Description

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Represents a message sent from the web application to the chatbot, containing metadata such as the current product page context.

b) Fields

Field Name	Type	Description
userId	string	The user who sent the message.
messageText	string	The content of the user's message.
imageUrl	string	Optional: image used for try-on or visual queries.
metadata	Map<string, any>	Contextual data (e.g., current productId, page type).

4.7.9. ChatResponse

a) Description

Represents the final response returned to the web UI, containing both natural language output and optional tool execution metadata.

b) Fields

Field Name	Type	Description
replyText	string	The chatbot's natural language response.
action	ClassifiedAction	Optional tool execution details for other system components.
errorMessage	string	Error message if processing fails.

4.8 Component: External Service

4.8.1 LLM (groq)

Description

The system integrates with Groq LLM through the official Groq SDK.

No direct HTTP calls or manual endpoint construction are required.

Instead, the backend invokes high-level SDK methods, and the SDK internally communicates with Groq's REST API endpoints.

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

Method Name	Functionality	Endpoint (SDK-Managed)
generateChatCompletion	Chat response generation	/openai/v1/chat/completions
callToolWithLLM	Tool/function calling	/openai/v1/chat/completions
generateEmbedding	Embedding generation (RAG)	/openai/v1/embeddings

4.8.2 Momo

Description

We integrate the MoMo Gateway to facilitate payments via versatile methods such as Credit Cards, ATM Cards (Napas), and QR Codes. The fundamental flow allows users to initiate a payment on our platform, be redirected to MoMo to complete the transaction, and seamlessly return.

Basic Flow:

- **Initiation:** The server sends a request to MoMo to generate a payment link (PayURL).
- **User Action:** The user pays via the MoMo App or Gateway.
- **Notification:** MoMo sends a Webhook (IPN) to our server to notify us of the result.
- **Confirmation:** Our server checks the inventory/database.
 - If valid: We send a **Capture** request to finalize the transaction.
 - If invalid (old payurl, cancel order): We send a **Cancel** request to release the hold on the user's funds.
- **Refund:** If a completed order is returned, we send a **Refund** request.

Key Endpoints (Server to MoMo)

Method Name	Functionality
requestPayUrl(orderInfo)	<p>Endpoint: Payment Creation</p> <p>Sends a request to /v2/gateway/api/create.</p> <p>It packages the order amount, order ID, and callback URLs. MoMo returns a payUrl (for web) and deepLink (for mobile) which we use to redirect the user to the payment screen.</p>

AI4STYLE	Version: 1.0
Software Architecture Document	Date: 27/11/2025
SAD-A4S	

confirmTransaction (orderId, type)	<p>Endpoint: Manual Confirm</p> <p>Sends a request to /v2/gateway/api/confirm.</p> <p>This is the critical "Decision" endpoint. We send a <code>requestType</code> of either "capture" (to complete the payment and take the money) or "cancel" (to void the transaction and release the money back to the user).</p>
refundTransaction (transId, amount)	<p>Endpoint: Refund</p> <p>Sends a request to /v2/gateway/api/refund.</p> <p>Used when an Admin approves a return or cancellation after the money has already been captured. It instructs MoMo to return a specific amount to the user's original payment method.</p>

5. Deployment

6. Implementation View