WINSHOP

Software Architecture Document

Abstract

This is a small project in our university course. The project is about a product that is an e-commerce website, a software product. The duration of this project is in the 1st semester of 2022

Nhap mon cong nghe phan mem 20_3

Team 05

GROUP 05

Winshop
Software Architecture Document

Version 1.1

Revision History

Date	Version	Description	Author
27/11/2022	1.0	Preliminary version	Group 05
10/12/2022	1.1	Update SAD	Group 05

Table of Contents

Software Architecture Document

1. Introduction

This Winshop project is built to create opportunities for everyone who needs to shop online, without wasting time searching for products. The products of the Winshop sold are completely guaranteed quality and the price is cheaper than the market price.
This document elaborates the software architecture document for "Winshop system" The system architecture is abstracted into many views and components which are explained in detail. The document follows the 4+1 view models as the reference model for this document.

1.1 Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of the Winshop application. It presents several different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system. In order to depict the software as accurately as possible, the structure of this document is based on the "4+1" model view of architecture. The "4+1" view model allows various stakeholders to find what they need in the software architecture.

1.2 Scope

The software architecture document applies to each static and dynamic aspect of the system. Since the 4+1 view model is used as the reference model, it corporates many views of the system, thus makes the document complete and consistent.
Under the static behavior of the system, the document discusses the class diagrams, package diagrams and other static architecture designs. Dynamic aspects of the system are elaborated using case realizations and system sequence diagrams.

1.3 Definition, Acronyms and Abbreviations

OOP - Object Oriented Programming
SAD - Software Architecture Document
API - Application Programming Interface

1.4 References

- Architectural Blueprints—The "4+1" View Model of Software Architecture, Philippe Kruchten, November 1995:
http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf

1.5 Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.
+ Section 2: Describes the architectural goals and constraints of the system
+ Section 3: Describes the use-case model of application
+ Section 4: Describes the architecture with components and relationships among them
+ Section 5: Describes how the system is deployed by mapping the components in Section 4 to machine running them
+ Section 6: Describes the folder structures for the code for all components described in Section 4

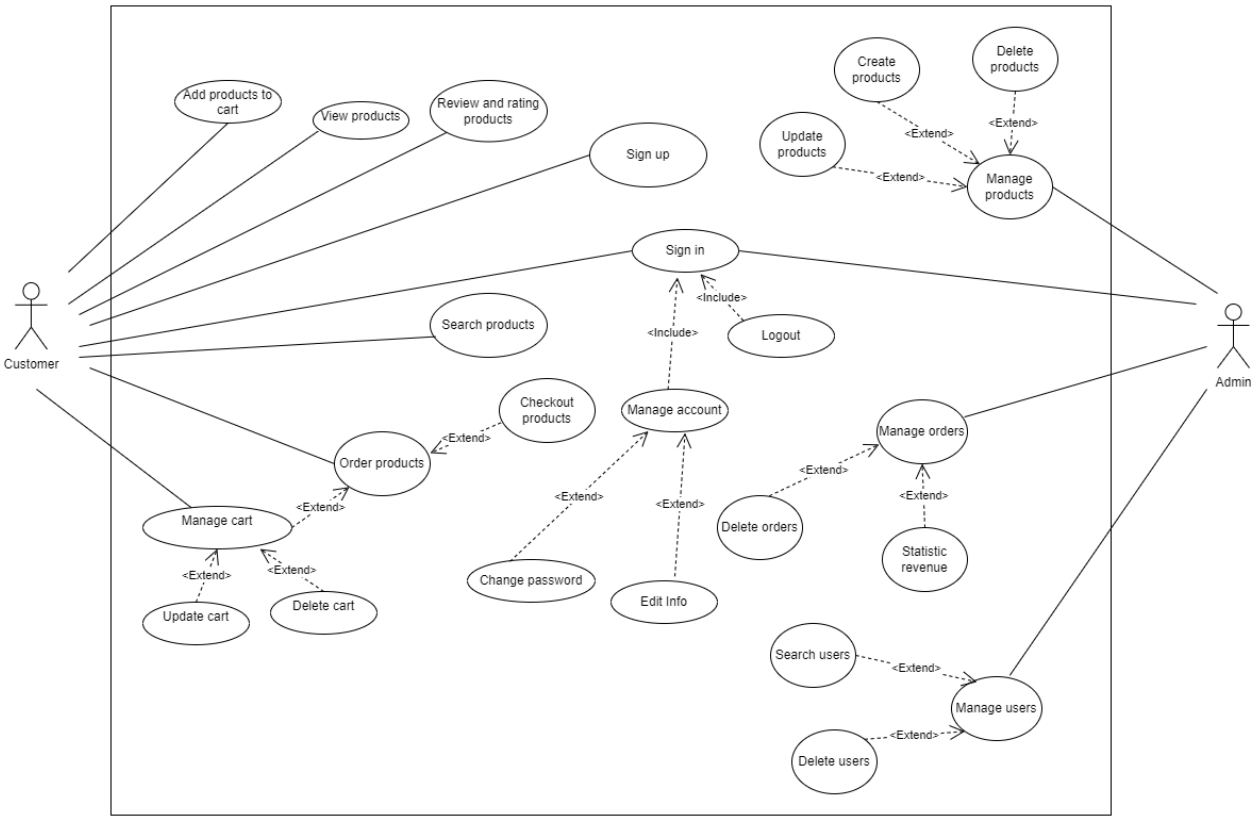
2. Architectural Goals and Constraints

- The system operates continuously 24/7.

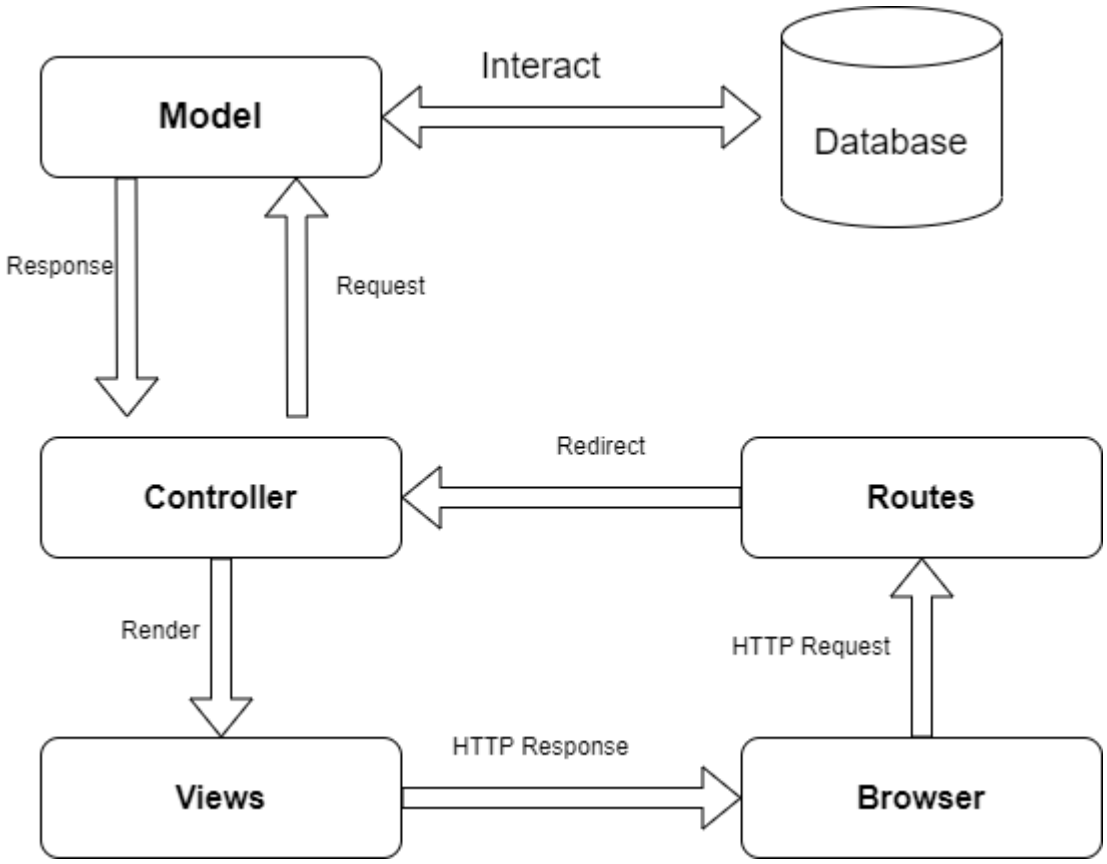
- WinShop will work on most operating systems
- User interface is friendly, easy to use for users.
- Customer information is confidential.
- All sensitive user data such as phone number, emails, password, or bank account are encrypted.
- Processing time does not exceed 5s.
- Supportability:
+ Platform: website
+ RAM: 1GB

3. Use-Case Model

Model



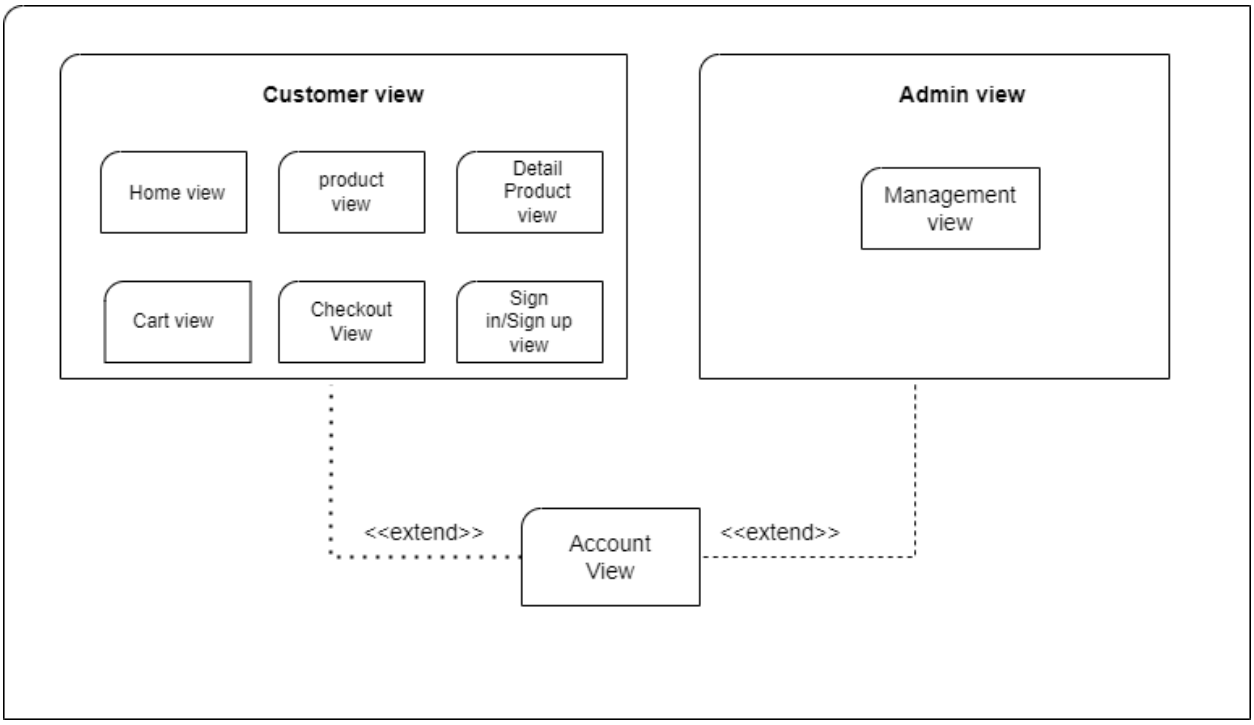
4. Logical View



- **Model:** Provide a data template, interact directly with the database to retrieve data. and response data for Controller component
- **Controller:** Handling logic according to user requests through view and interact with Model
- **Views:** Show the user interface to Browser
- **Routes:** When the user executes the event on the website, the browser will call the http request and route will redirect to appropriate controllers.
- **Database:** Using MongoDB this component's duty is to store all the information about the user and the system.

4.1 Component: Views

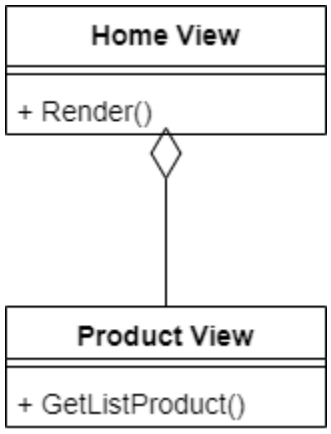
- The main goal of this component is to render and update the user interface (UI) and interact with controllers after the user's interaction with the website



Package diagram of Views

4.1.1 Home view

- **Responsibilities:** The main responsibility of this view is to display a list of current products, homepage of the website, user can also proceed to other views like account view, detail product view.
- **Class Diagram:**

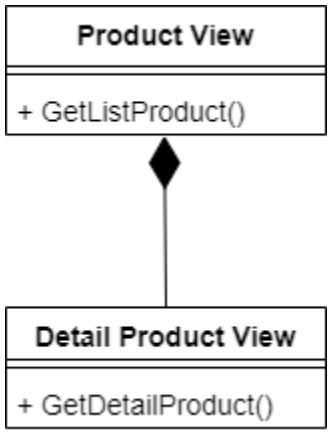


- **Explain:** Render () return interface for user. Function GetListProduct() returns a list of current products and displays the interface.

4.1.2 Product View

- **Responsibilities:** The main responsibility of this view is to show a list of products. When clicking on any product type, it will go to Detail Product View showing detailed information about each different product type.

- **Class Diagram:**



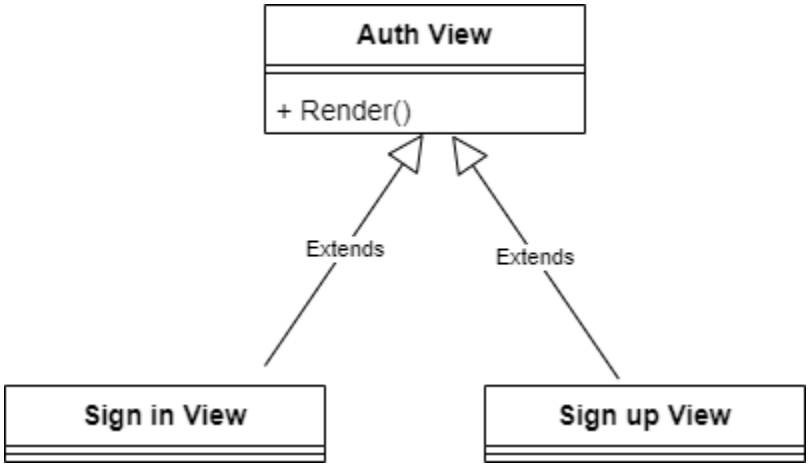
- **Explain:** Function GetListProduct () return list of current products and display to interface. And function GetDetailProduct () showing detailed information about each different product type.

4.1.3 Detail Product View

- **Responsibilities:** this view is to display detailed information about each different product type.
- **Class Diagram:**

4.1.4 Sign in/Sign up View

- **Responsibilities:** Auth View is a group of interfaces in charge of authenticating user data, so it consists of 2 child views: Sign in View and Sign-up View. The main responsibility of these views is to display an interface for user to create an account or log in with an existed account
- **Class Diagram:**

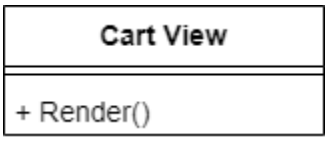


- **Explain:** Render () return interface for user

4.1.5 Cart View

- **Responsibilities**: Display the interface of products added to the cart by customers.

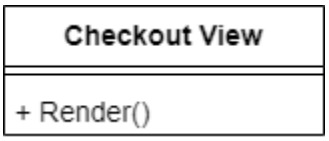
- **Class Diagram**:



- **Explain:** Render () return interface for user

4.1.6 Checkout view

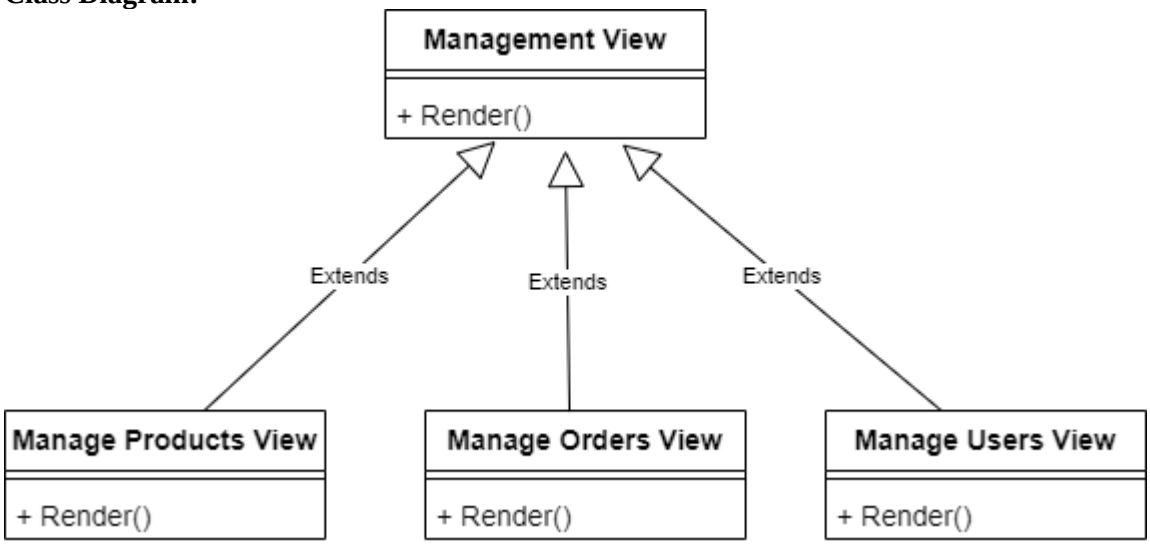
- **Responsibilities:** Display the user's payment information interface after ordering products
- **Class Diagram:**



- **Explain:** Render () return interface for user

4.1.7 Management View

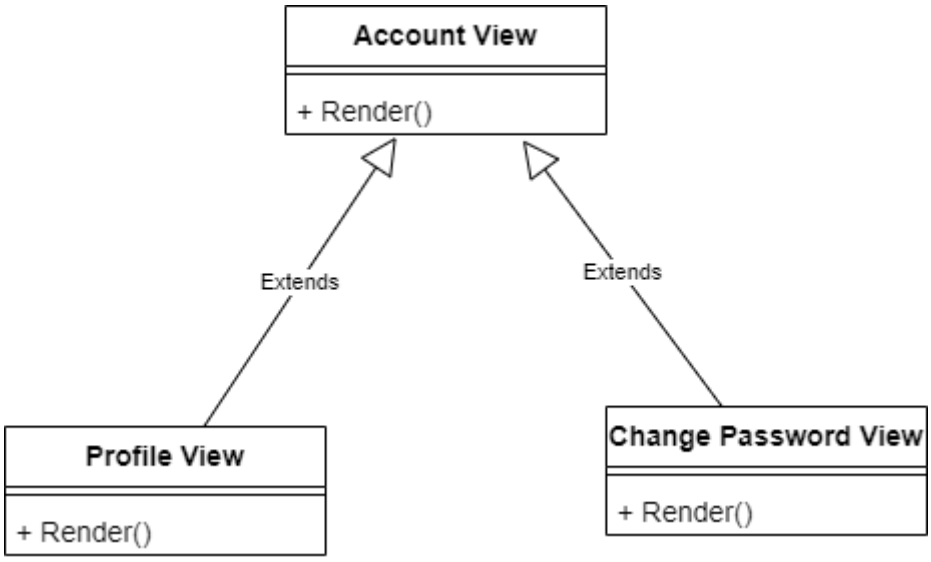
- **Responsibilities:** This view is the admin interface, which appears when the admin logs in to the admin page, allowing the admin to view product sales statistics, view and edit the list of products, orders, users.
- **Class Diagram:**



- **Explain:**
  + Render () return interface for user
  + Manage Products View display list of products to interface.
  + Manage Order View display list of Order to interface.
  + Manage User View display list of User to interface.

4.1.8 Account View

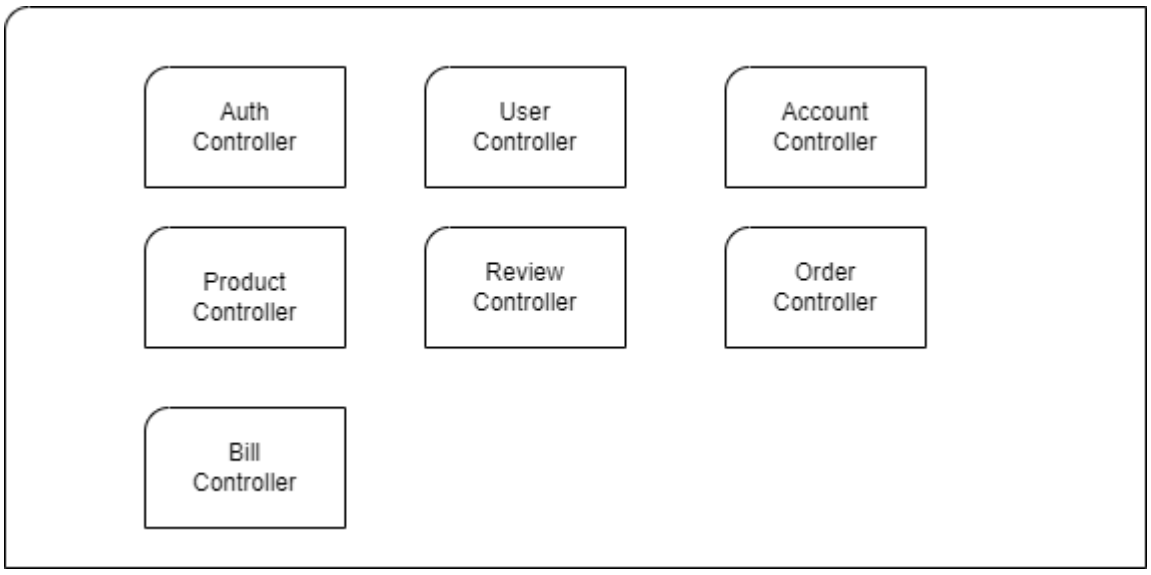
- **Responsibilities:** This view is the interface of each user's personal account, appearing when the user selects the user icon on the navigation bar, allowing the user to view and edit personal information, change the password.
- **Class Diagram:**



- **Explain:**
  + Render () return interface for user
  + Profile View displays user information.
  + Change password View displays interface to allow user to change password.

4.2 Component: Controllers

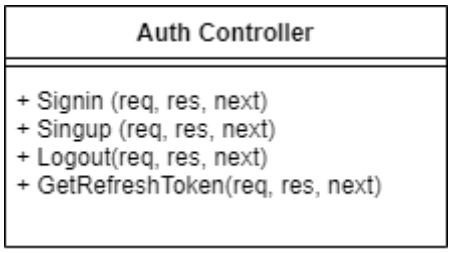
Controllers are the components that handle user interaction, work with the model, and ultimately select a view to render. The view only displays information; the controller handles and responds to user input and interaction



4.2.1 Auth Controller

- **Responsibilities:** The Auth controller is responsible for handling logic about registration, login, logout and refreshing the new token after a period.

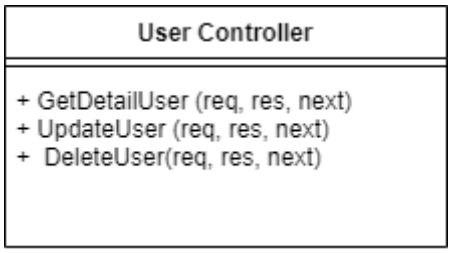
- **Class Diagram:**



- **Explain:**
  + Singup(): Make a sign up request
  + Signin(): Make a login request
  + Logout(): Make a logout request.
  + GetRefreshToken(): Make a request to create a new token

4.2.2 User Controller

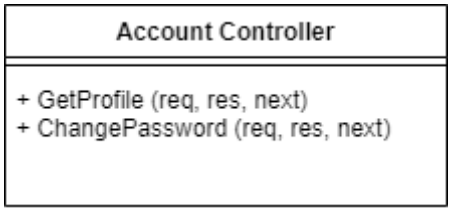
- **Responsibilities**: The user controller is responsible for handling the logic of user management request
- **Class Diagram:**



- **Explain:**
  + GetDetailUser(): Make a request to get user information
  + UpdateUser(): Make a request to update user information.
  + DeleteUser(): Make a request to delete user information.

4.2.3 Account Controller

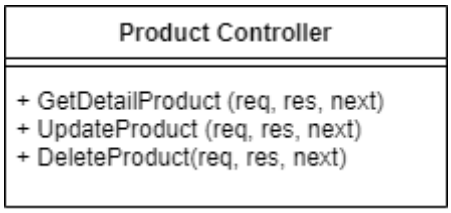
- **Responsibilities:** Handle user requests on personal pages such as viewing personal information, changing passwords
- **Class Diagram:**



- **Explain:** GetProfile() makes a request to get a profile. ChangePassword() makes a request to change password.

4.2.4 Product Controller

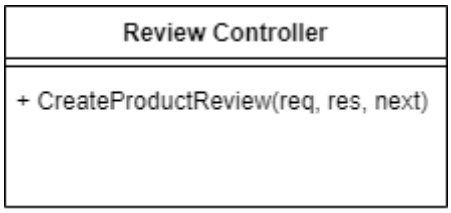
- **Responsibilities:** The product controller is responsible for handling the logic of product management request
- **Class Diagram:**



- **Explain:**
  + GetDetailProduct(): Make a request to get product information.
  + UpdateProduct(): Make a request to update a product.
  + DeleteProduct(): Make a request to delete a product.

4.2.5 Review Controller

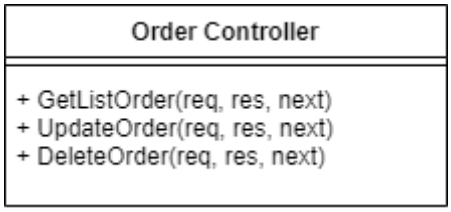
- **Responsibilities:** Handling the logic of adding comments to products.
- **Class Diagram:**



- **Explain:** CreateProductReview(): Make a request to create a review of a product.

4.2.6 Order Controller

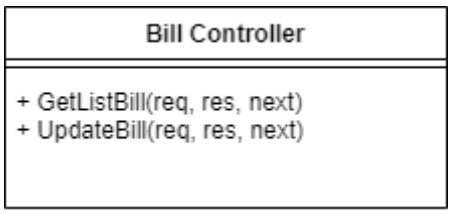
- **Responsibilities:** Perform tasks to get order list, delete order, update order
- **Class Diagram:**



- **Explain:**
  + GetListOrder(): make a request to get a list of orders.
  + UpdateOrder(): make a request to update order.
  + DeleteOrder(): make a request to delete order

4.2.7 Bill Controller

- **Responsibilities:** Handle admin requests such as viewing invoice list or editing
- **Class Diagram:**

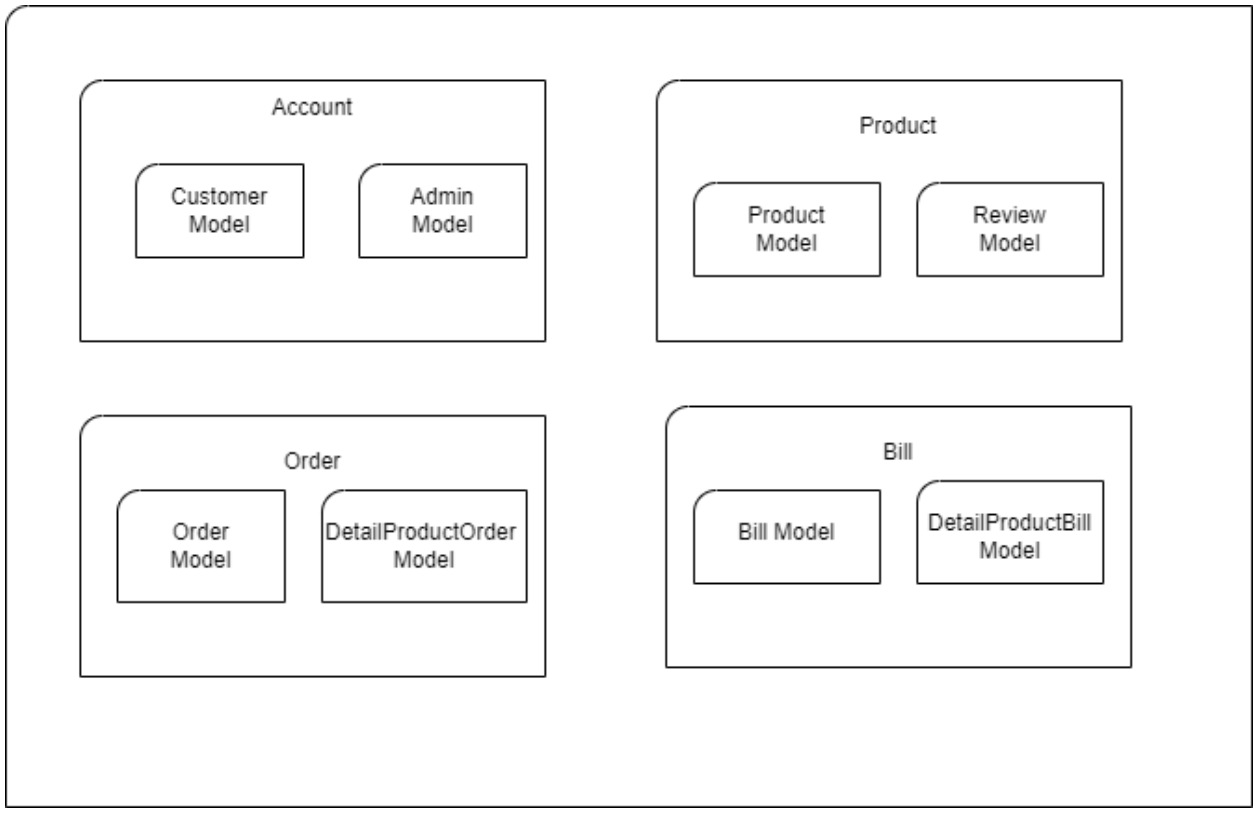


- **Explain:**
  + GetListBill(): Make a request to get a list of bills.

+ UpdateBill(): Make a request to update bills.

4.3 Component: Models

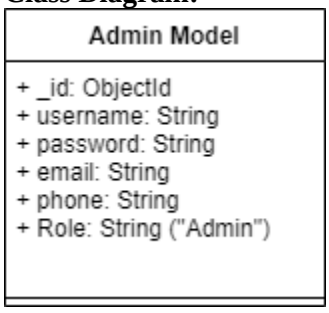
The model class represents the provision of a data template, interacting directly with the database to retrieve data.

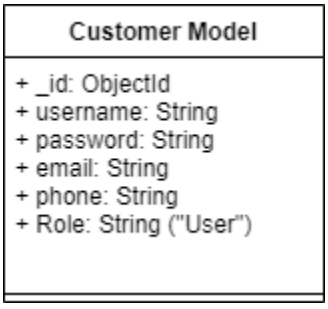


Package Diagram of Model

4.3.1 Account Model

- **Responsibilities:** Class includes properties containing account information
- **Class Diagram:**

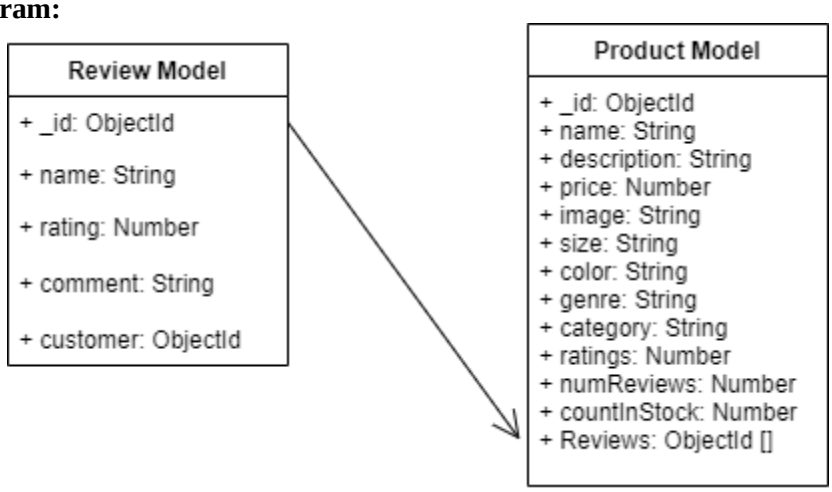




4.3.2 Product Model

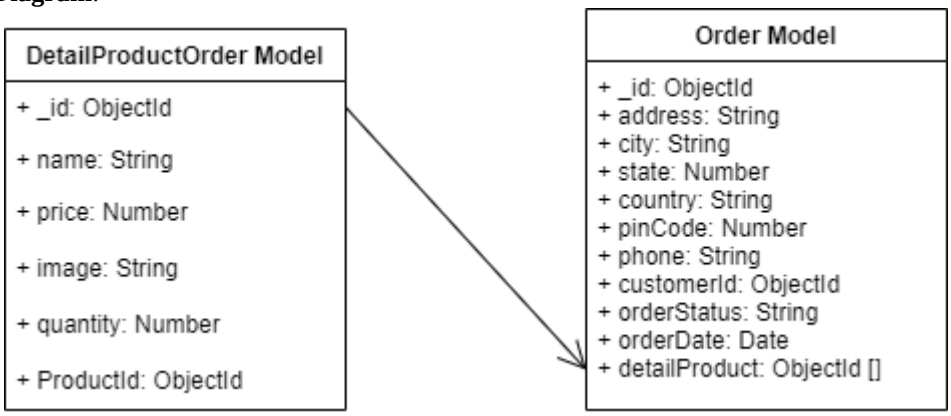
- **Responsibilities:** Class includes attributes that are product information and product reviews

- **Class Diagram:**



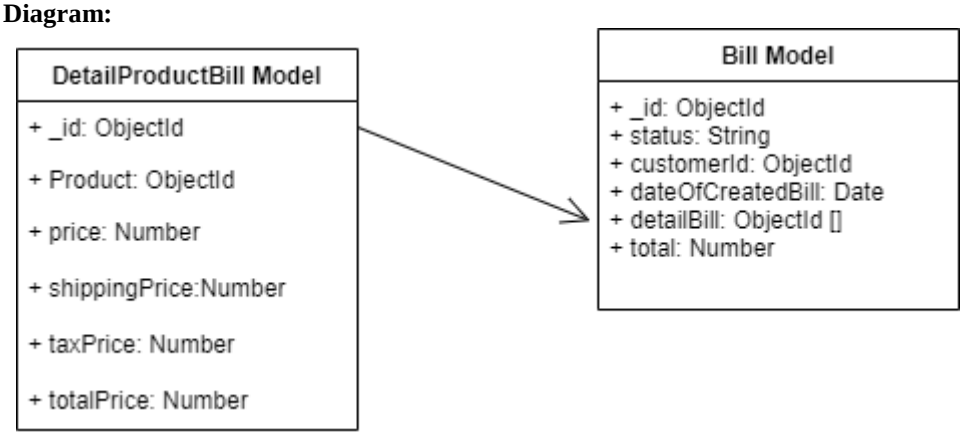
4.3.3 Order Model

- **Responsibilities**: Class contains attributes that are information about orders
- **Class Diagram**:



4.3.4 Bill Model

- **Responsibilities:** Class includes attributes are the information of the invoice
- **Class Diagram:**



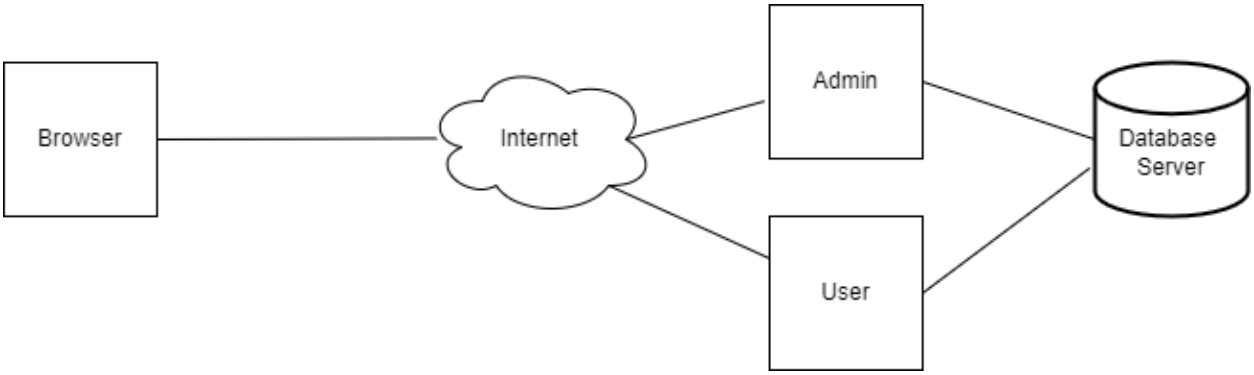
4.4 Component: Routes

- **Responsibilities:** Route defines the URL pattern and handler information. All the configured routes of an application are stored in the Route Table and will be used by the Routing engine to determine appropriate handler class for a request.

4.5 Component: Database

- **Responsibilities**: Using MongoDB, store all the data about the website, including user account, products, orders, …

5. Deployment



- **Browser:** The client will access the web through the browser and send requests to the servers via the internet.
- **Admin:** Receive request from browser (admin object), define corresponding routes to pass to controller. Handling logic according to user requests and interact with model to get data from the database.
- **User:** Receive request from browser (user object), define corresponding routes to pass to controller. Handling logic according to user requests and interact with model to get data from

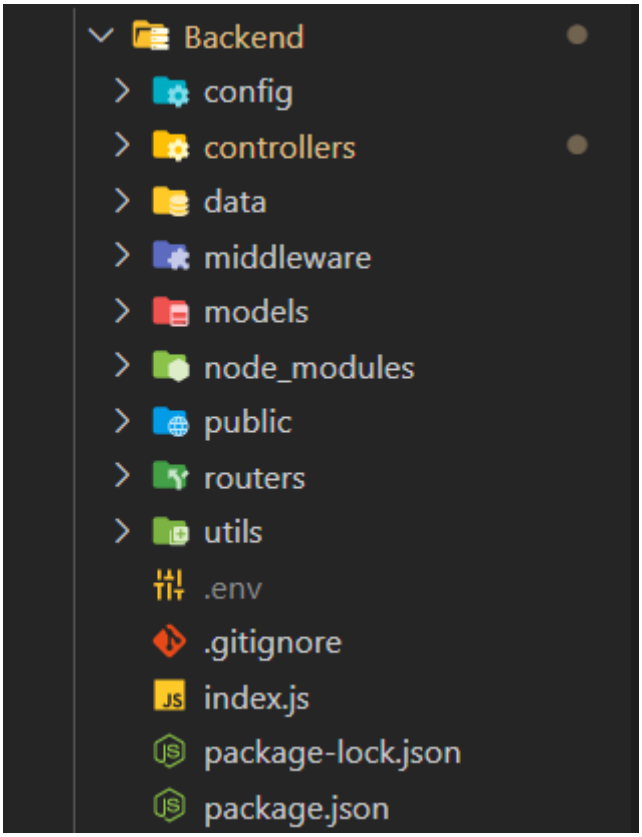
the database.

- **Database**: Store data of the application

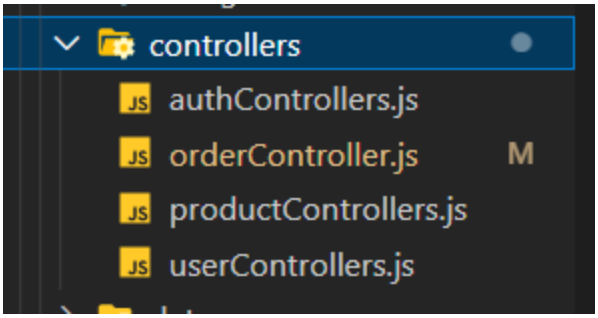
6. Implementation View

6.1 Project Backend:

- Constructor directory:

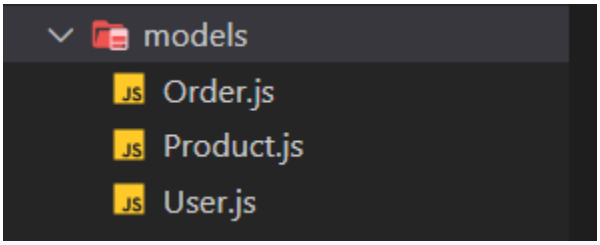


- **Folder config:** contains files to backup data in case the data on the database is corrupted
- **Folder controllers:**

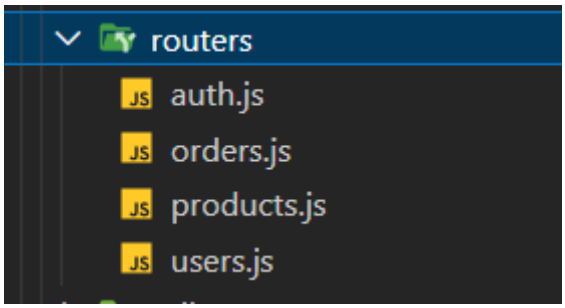


- Contain files to receive the request from the browser that the route navigates to and receives the result from the model returned and then renders the view to the browser.

- **Folder data:** Contains data files that need to back up
- **Folder Middleware:** Contains files that act as an intermediary between request/response to control errors and handle logic inside the web server.
- **Folder models:**



  - Contains files that define the database structure and related settings.
- **Folder routes:**

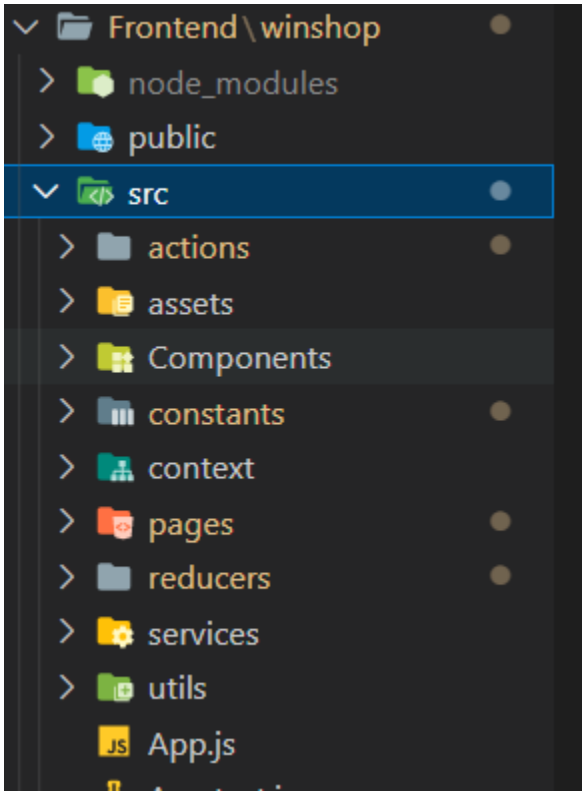


  - Contains files that direct requests to the respective controllers.
- **Folder public:** contains files public

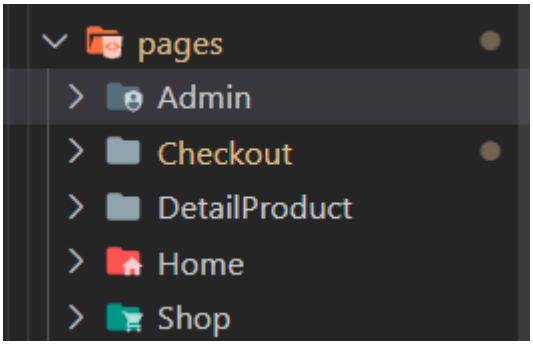
6.2 Project Frontend:

- Frontend acts as a component view of the application

- Constructor directory:



- **Folder Components:** Contains the components required for an application page
- **Folder pages:**



  - Contains js files that make up an application page, including admin and user pages
- **Folder constants:** Contains application constants
- **Folder reducer:** Contains global variables and is widely used for different components
- **Folder utils:** Contains functions that are reused many times
- **Folder services:** Contains functions to connect the server to the browser and render the user interface