EFOOD

Software Architecture Document

Version <1.4>

Revision History

| **Date** | **Version** | **Description** | **Author** |
| --- | --- | --- | --- |
| 28/11/2022 | 1.0 | Home Component, Payment Component, User profile Component, User controller | Trần Anh Khôi |
| 29/11/2022 | 1.1 | Logic View, Sign in Component, Sign up Component, Authentication controller, ReactJS web application architecture | Trần Dũng Tiến |
| 7/11/2022 | 1.2 | Cart controller, Order controller, User service, Product service, Order service | Nguyễn Trung Kiên |
| 7/11/2022 | 1.3 | Use case Model, Product Component, Shopping cart service | Đinh Cao Hồng Phước |
| 8/11/2022 | 1.4 | Admin Component, Admin controller, Server Architecture | Trần Bảo Long |

Table of Contents

1. Introduction 4

2. Architectural Goals and Constraints 4

3. Use-Case Model 4

4. Logical View 4

5. Deployment 4

6. Implementation View 4

Software Architecture Document

# Introduction

**Purpose:**

* This document provides a thorough architectural overview of the system, using several different architectural views to demonstrate different aspects of the system. It models the important architectures that have been implemented on the system and how use cases are structured in the architecture.

**Scope:**

* This document will define a high-level design and technology decision of the system. In this document, we focus on the choices made for The Coffee Factory system.

**References:**

* Vision document
* Use-case model
* Use-case specification

**Definitions, Acronyms, and Abbreviations:** None

**Overview**

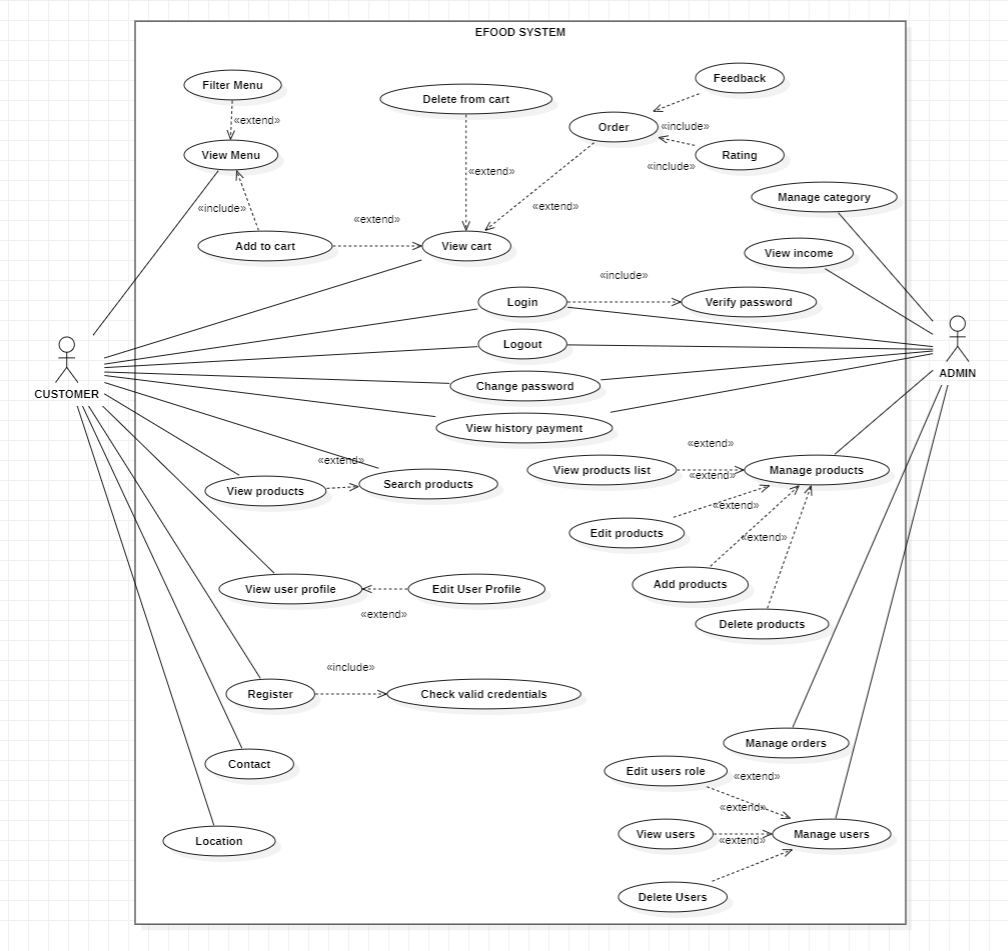
* The software architecture document is divided into these sections:

1. Introduction
2. Architectural goal and constraints
3. Use-case model
4. Logical overview
5. Deployment
6. Implementation view

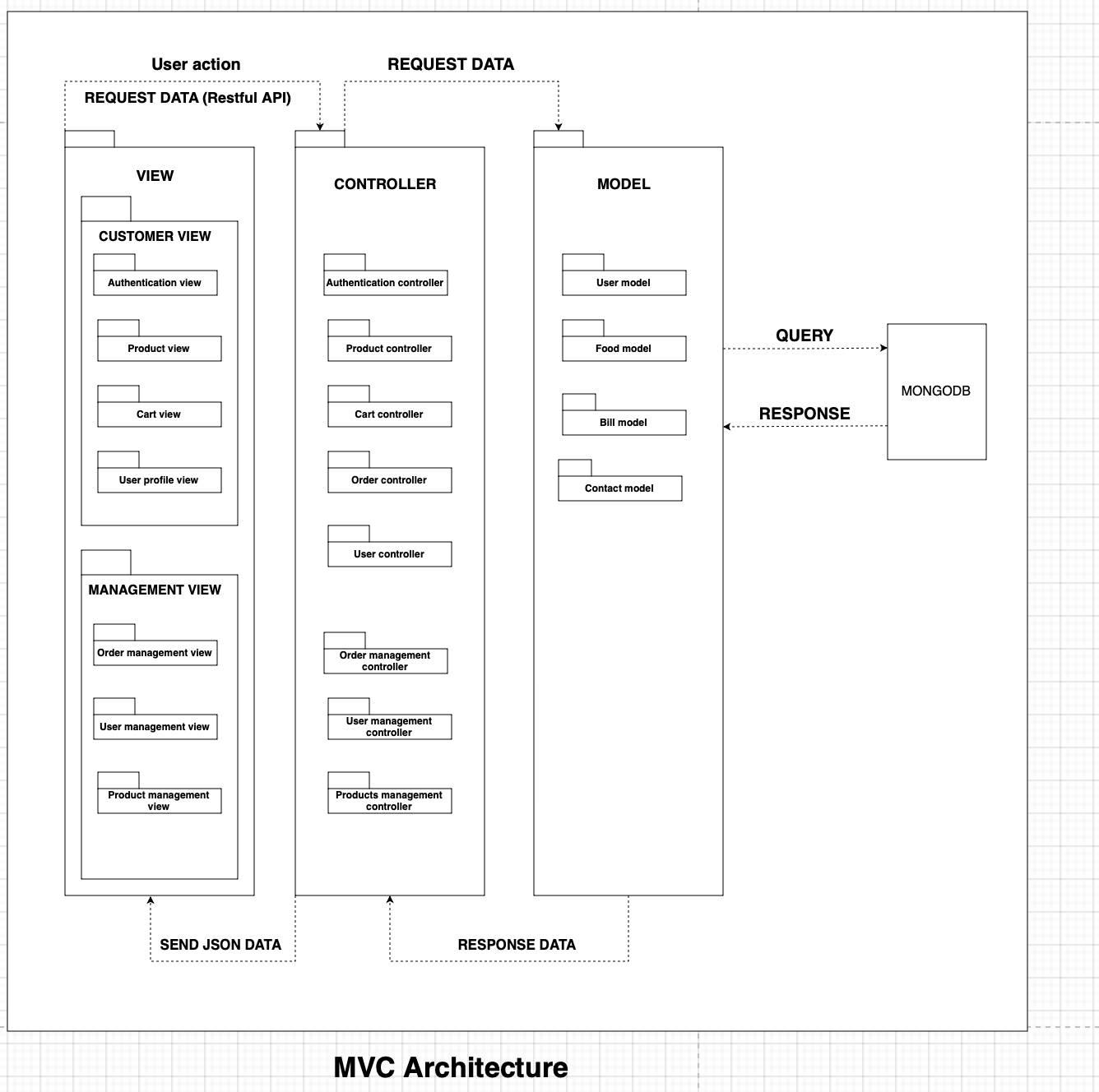
# Architectural Goals and Constraints

* **Key requirements:**
* User has to log in to perform protected actions.
* The homepage has to recommend best-seller products, and favorite products.
* Payment with 2 methods: Momo wallet, on delivery.
* The system is a MERN stack web application, the server will be deployed on <**SERVER>** connected to the MongoDB Atlas database.
* **Applicable standards:**
* The user-interface has a specific version of window, or mobile
* Easy to use, easy to interact with.
* Good UX/UI.
* **Server System requirements**
* Database is stored in mongoDB Atlas. - Using the Express framework.

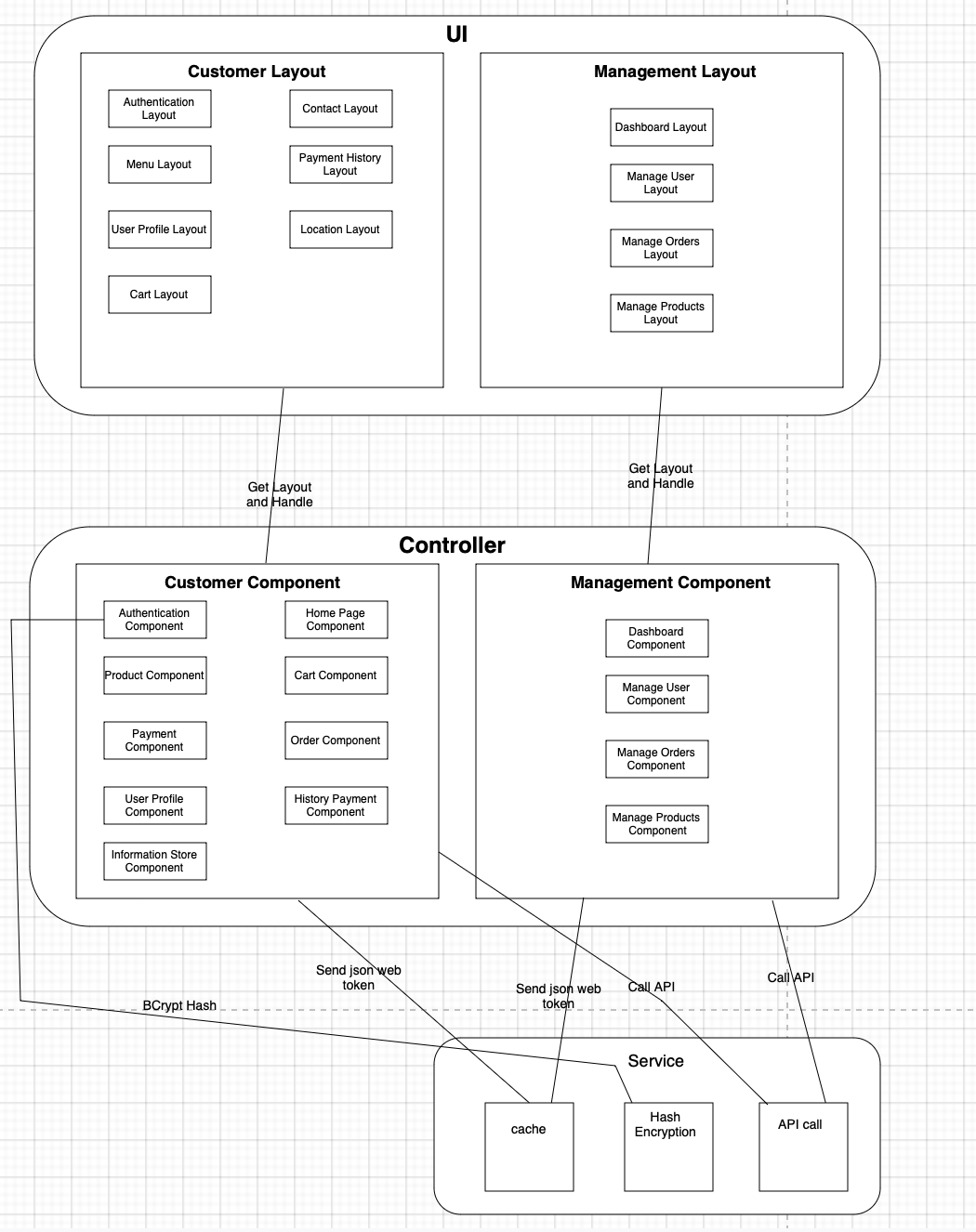
# Use-Case Model



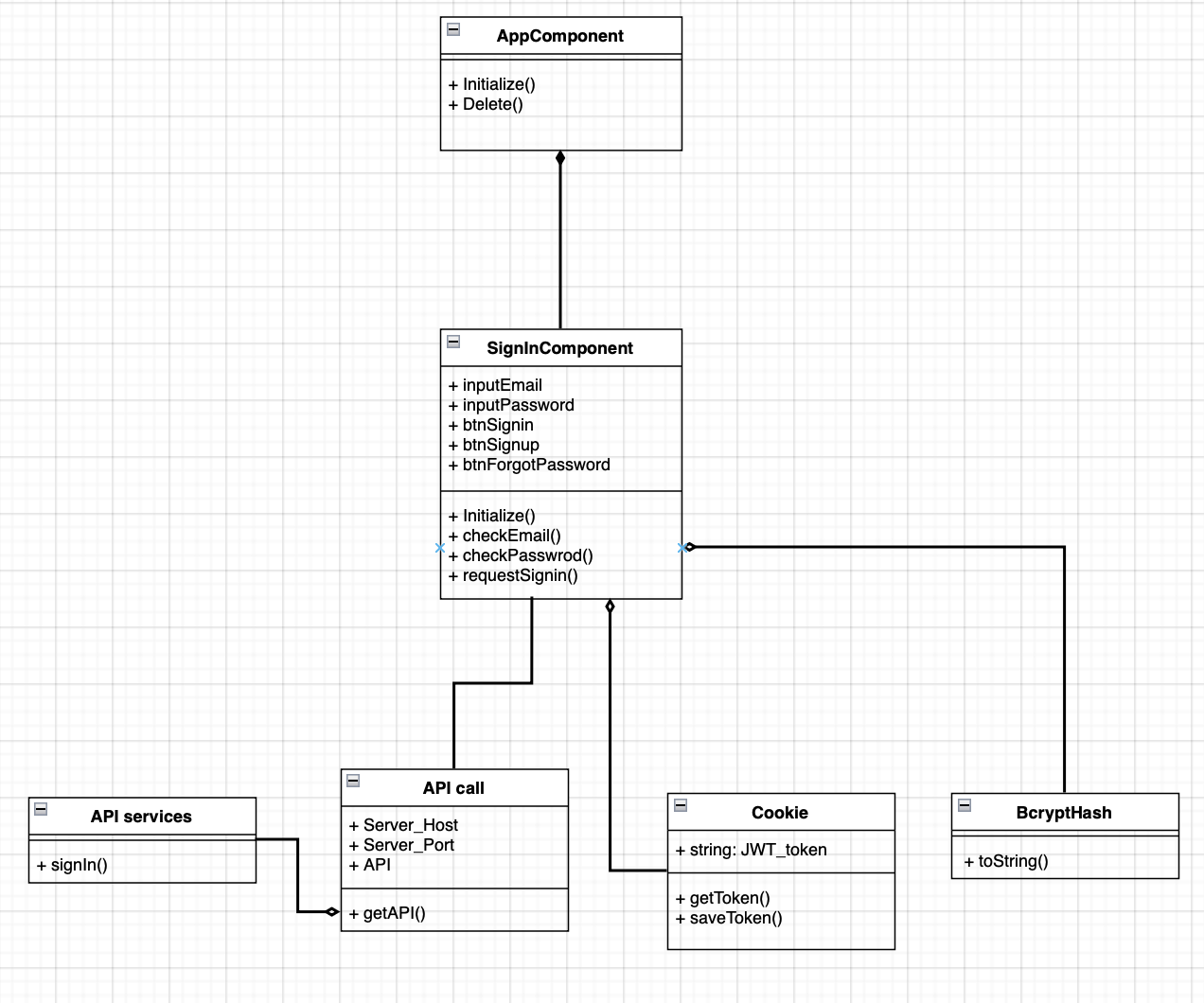
# Logical View

**

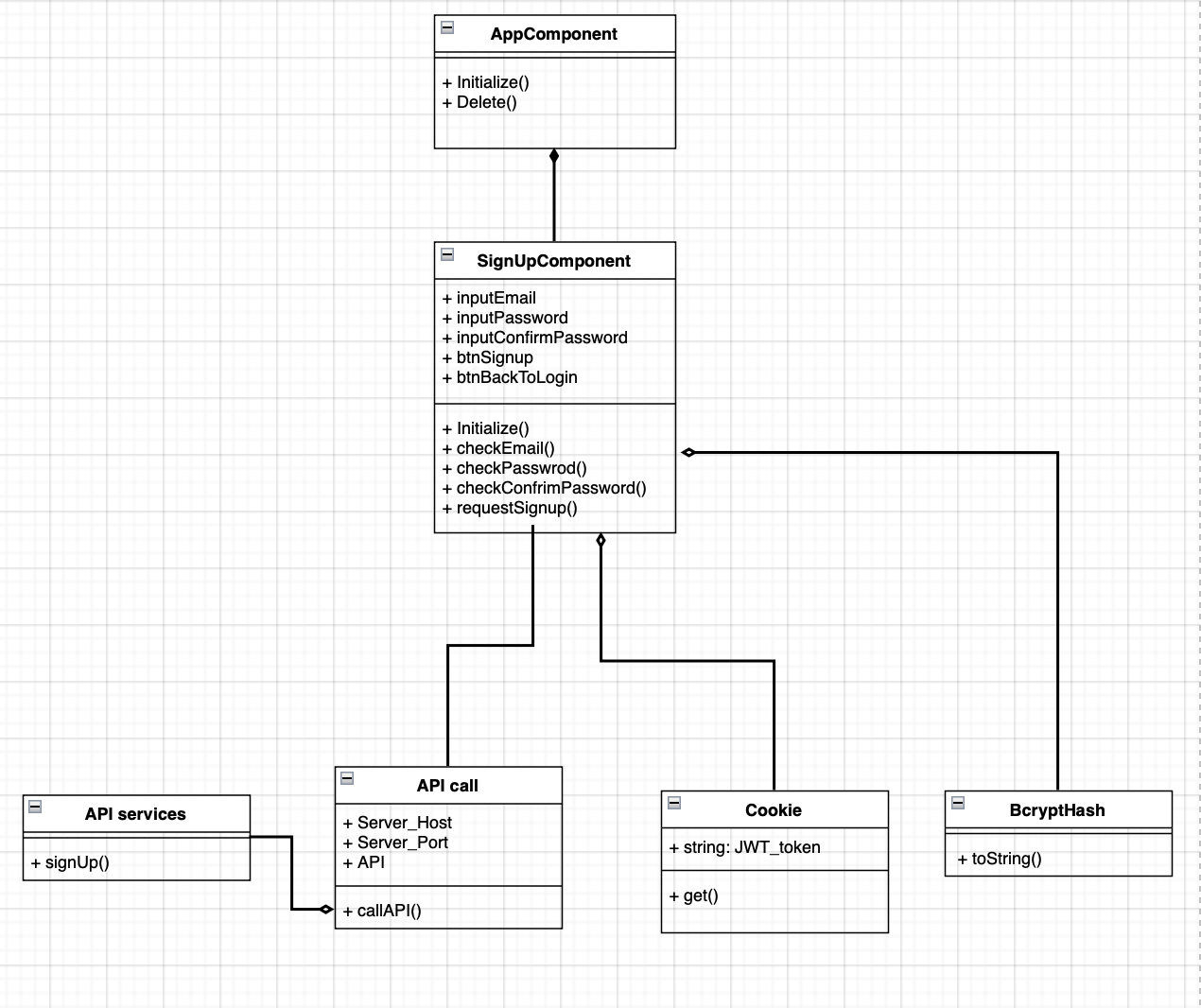
## ReactJS web application architecture



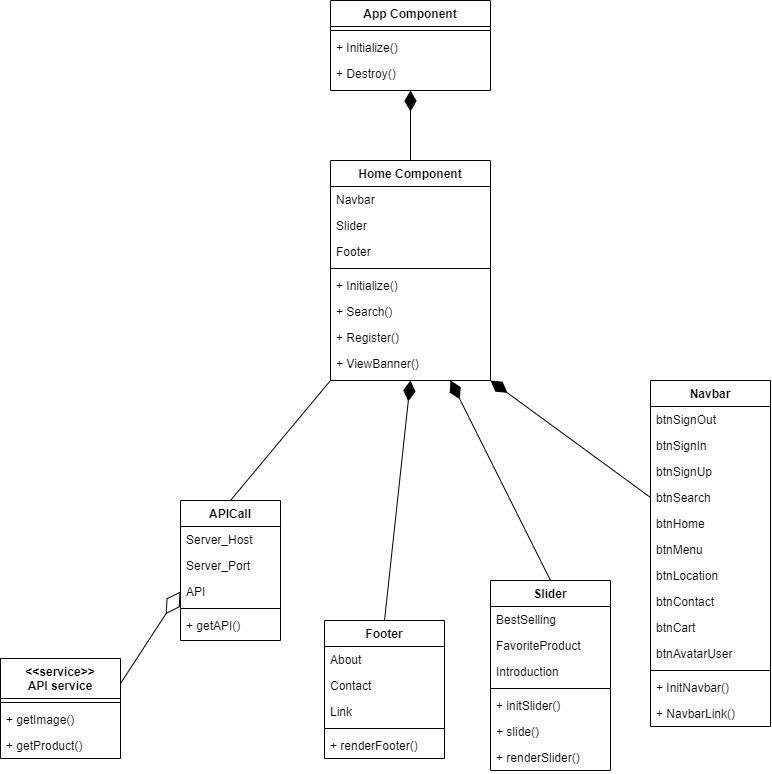
* + 1. **Sign in Component**

****

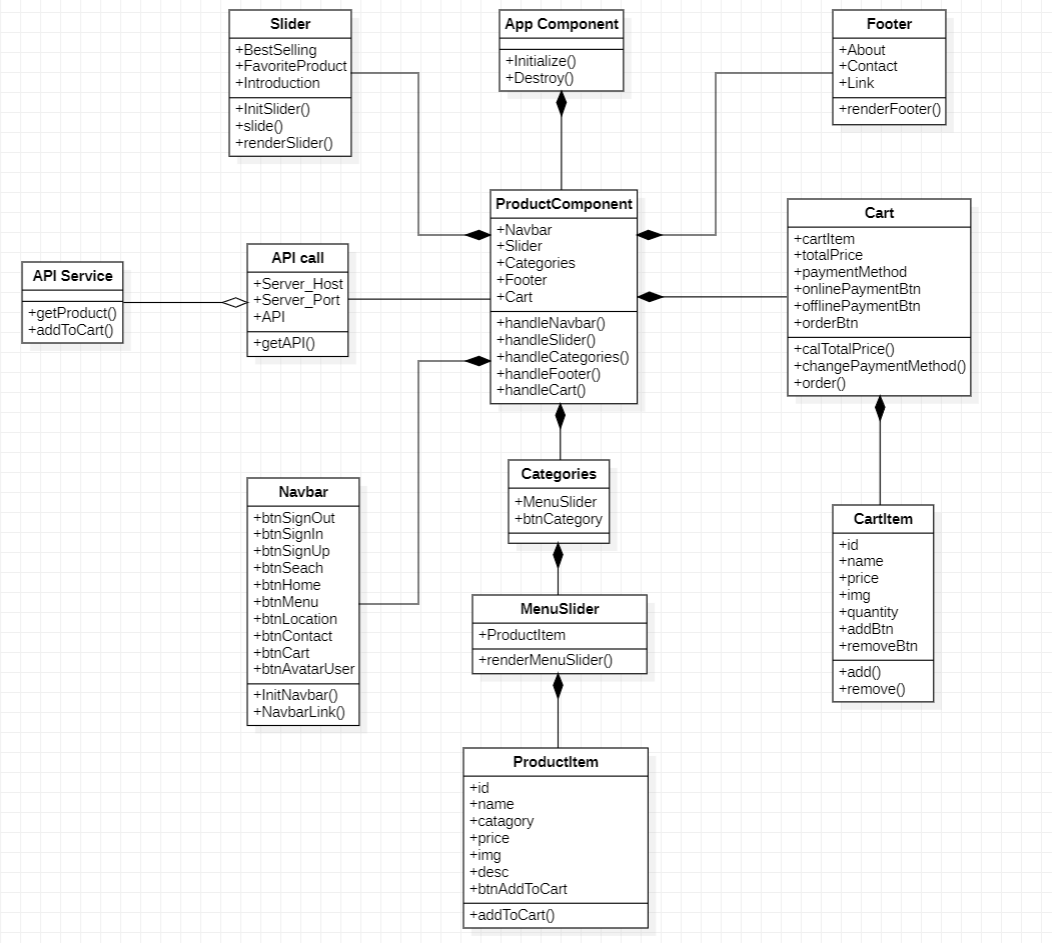
* + 1. **Sign up Component**

****

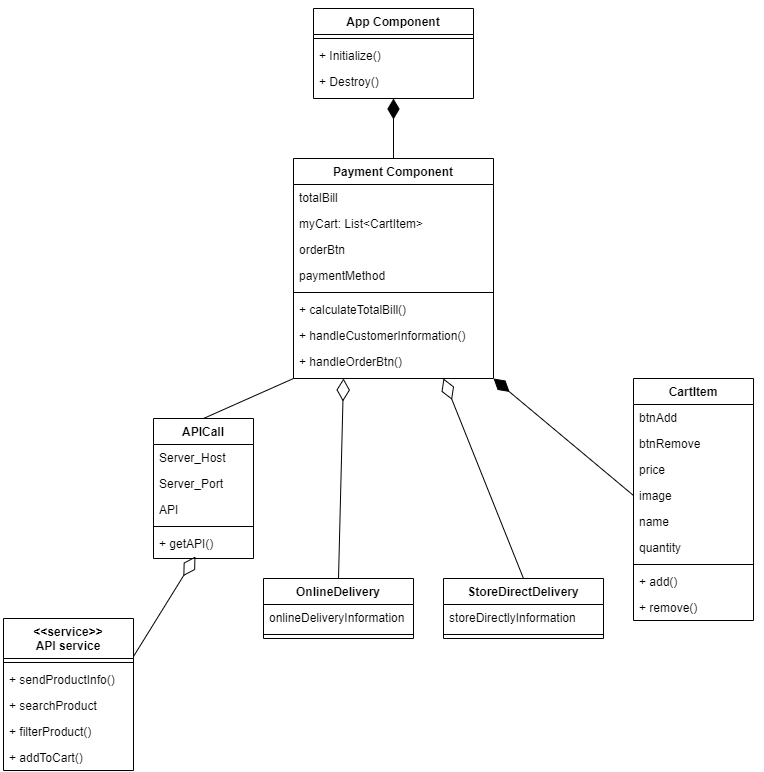
* + 1. **Home Component**

****

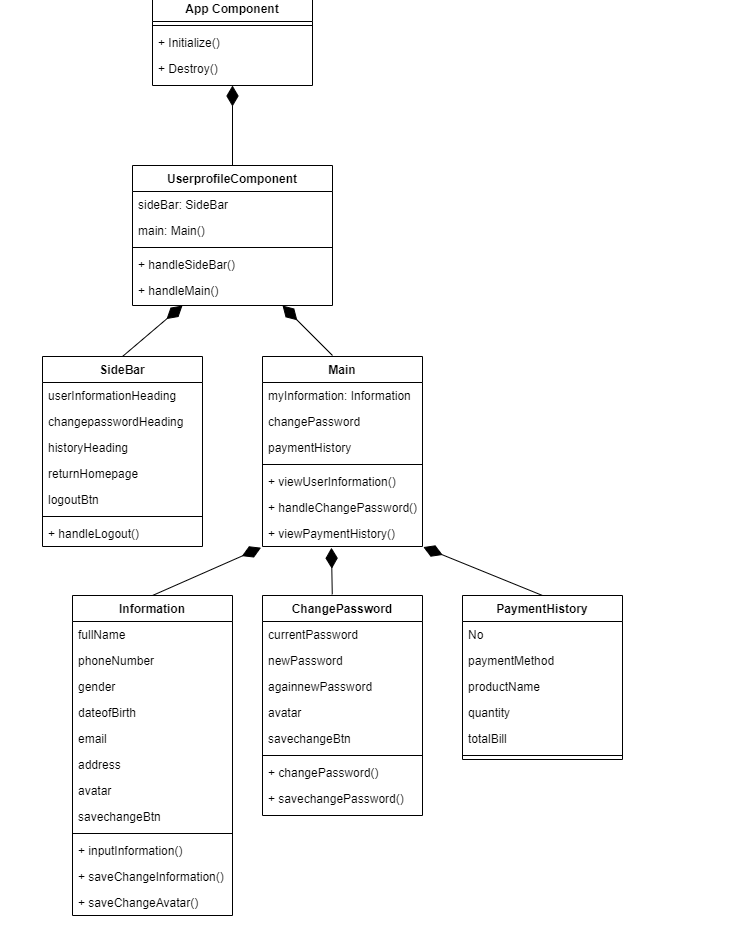
* + 1. **Product Component**

****

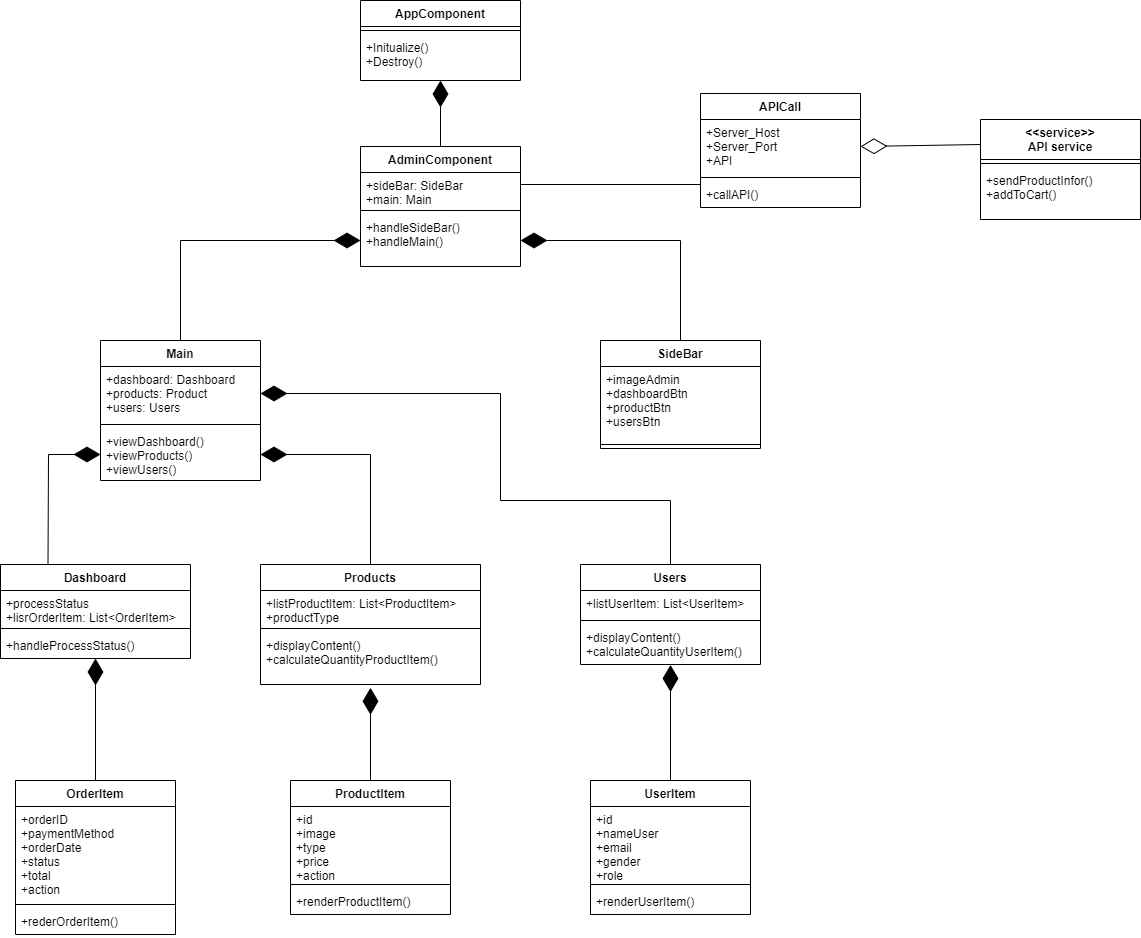
* + 1. **Payment Component**

****

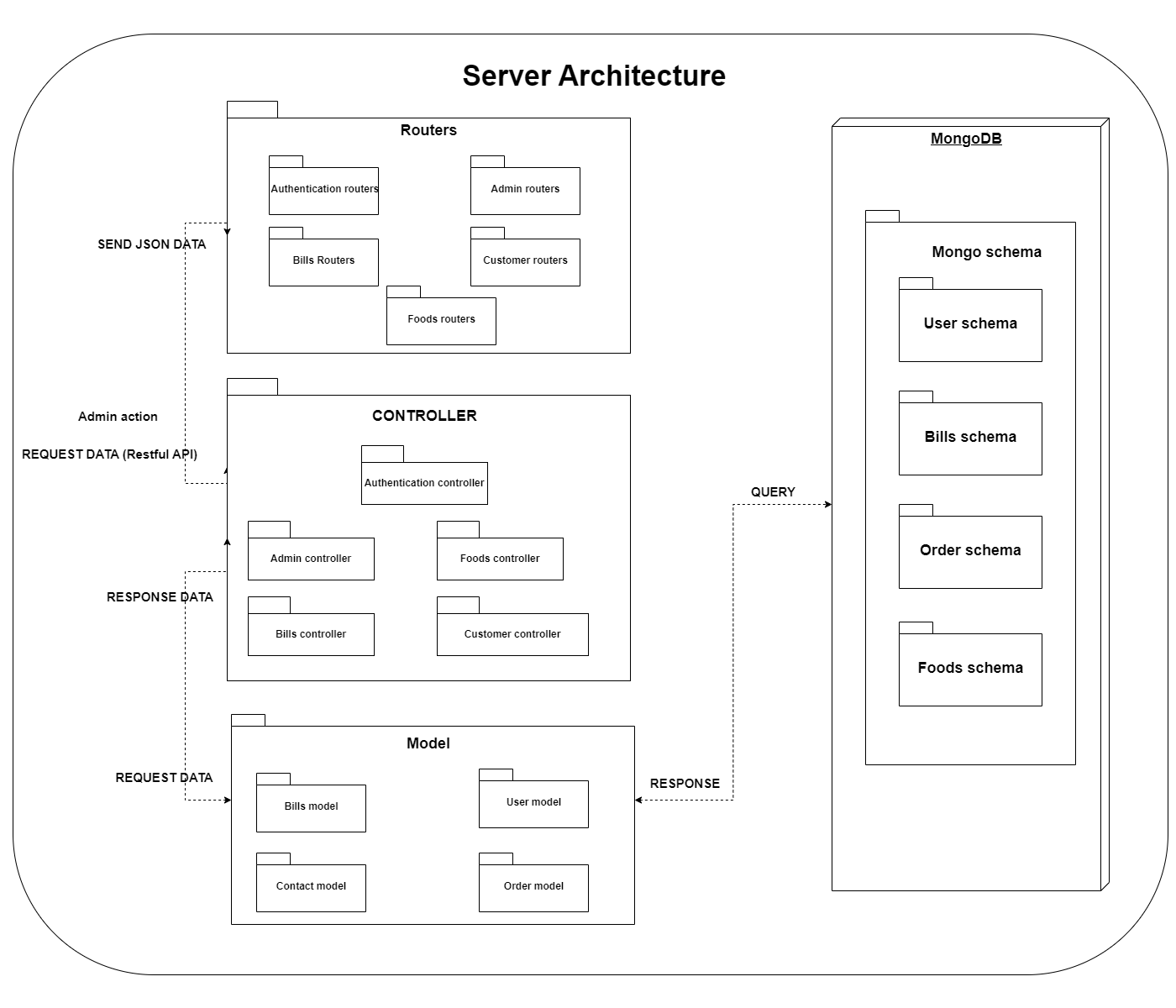
* + 1. **User profile Component**

****

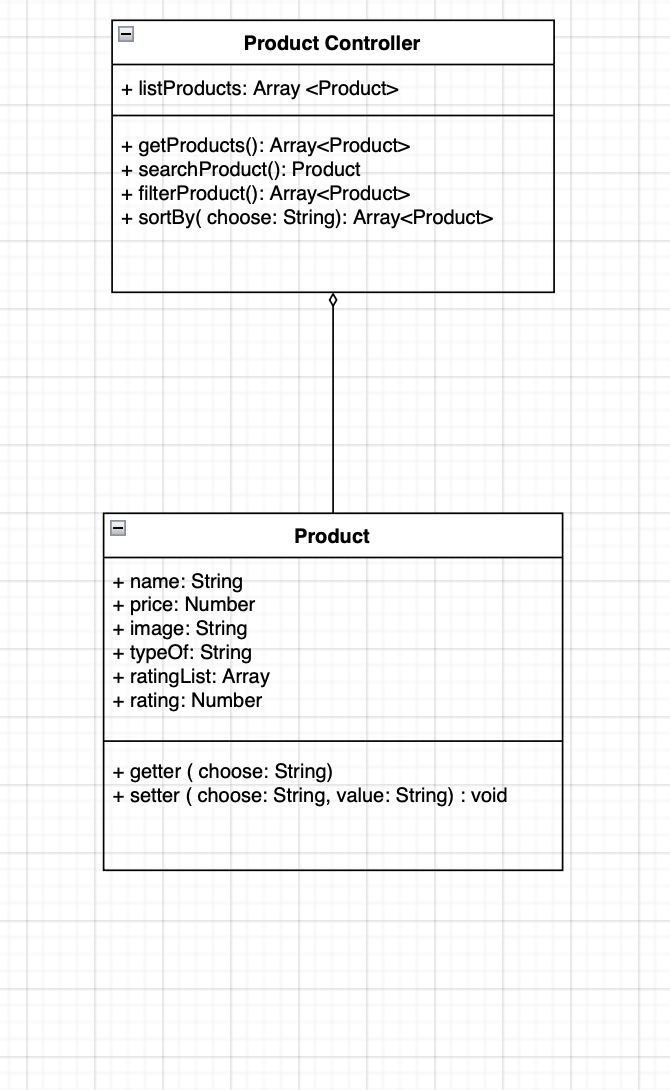
* + 1. **Admin Component**

****

* 1. **Server Architecture Long**

****

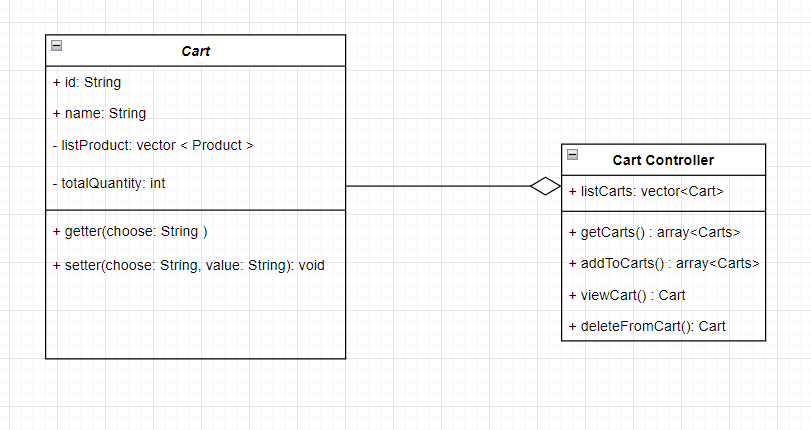
* + 1. **Product controller**

****

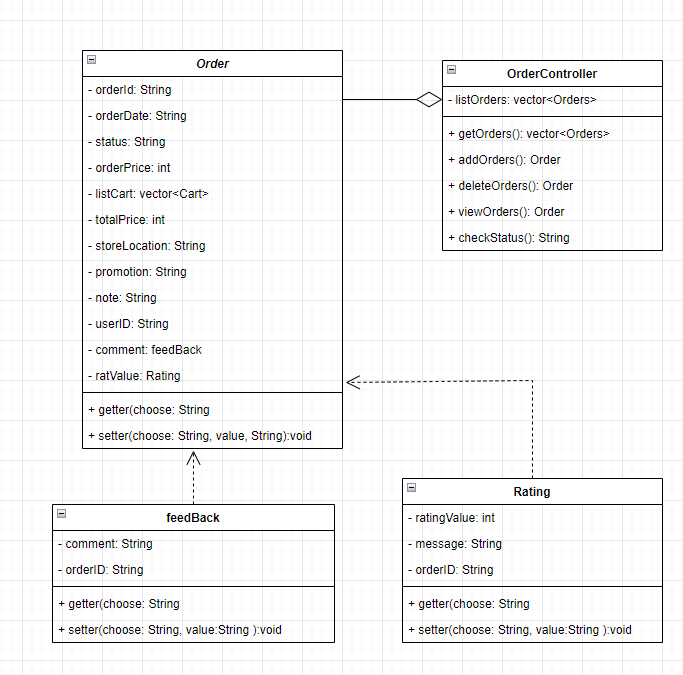
At Product component, it has 2 methods namely getter and setter. Instead it has 5 get methods and 5 set methods such as getName(), getPrice(), setName(name),.... We merge it into getter(choose) and setter(choose, value) with parameter choose to select properties that the developer wants to interact with.

Example:

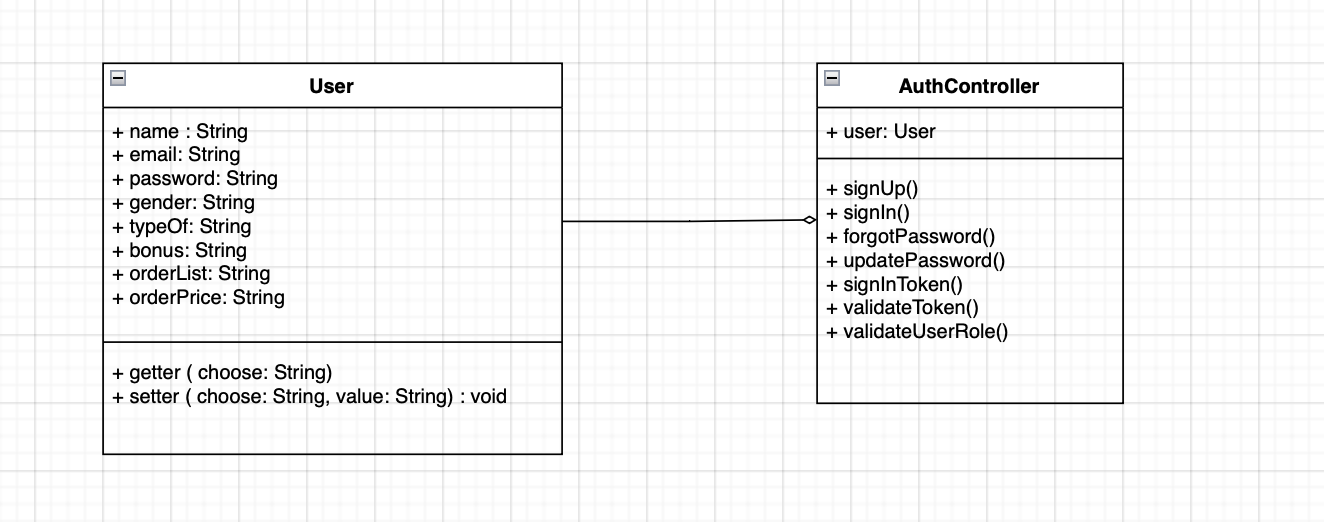
* getter(“name”) is similar to getName() that returns name of Product
* getter(“price”) is similar to getPrice() that returns price of Product
* setter(“name”, “newName”) is similar to setName(“newName”) that sets value of name by “newName”
  + 1. **Cart controller**

****

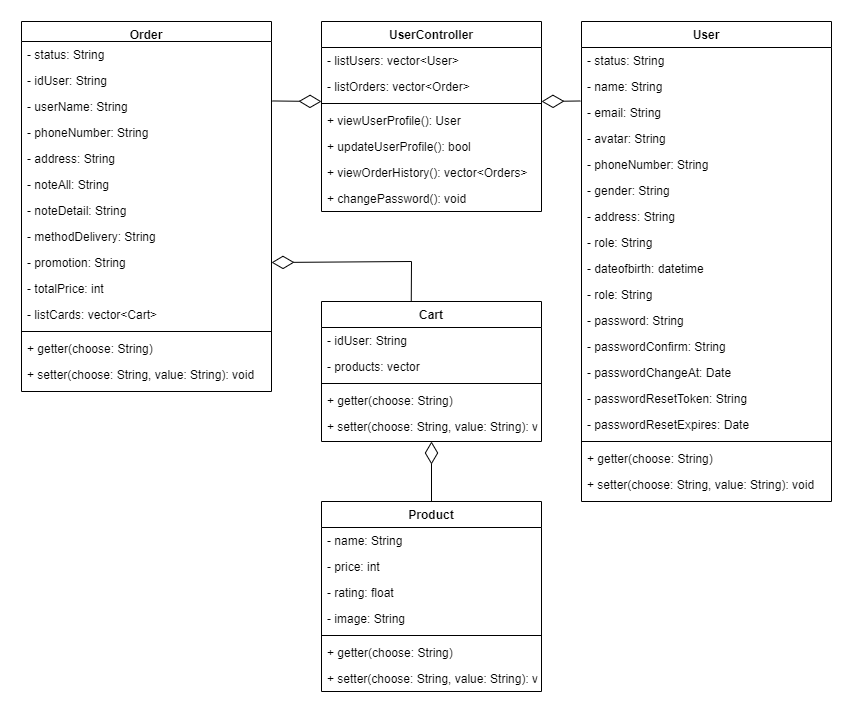
* + 1. **Order controller**

****

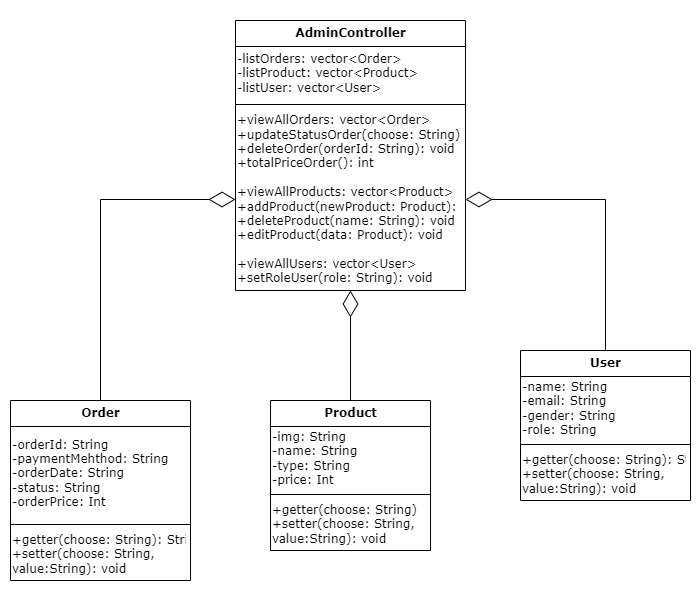
* + 1. **Authentication controller**

****

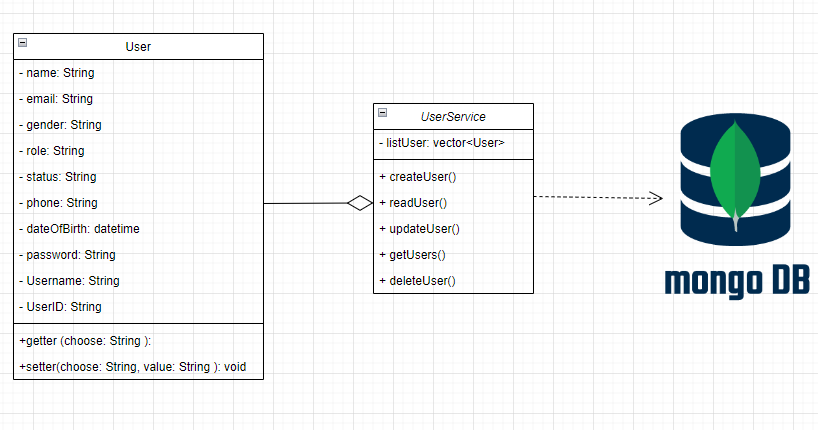
* + 1. **User controller**

****

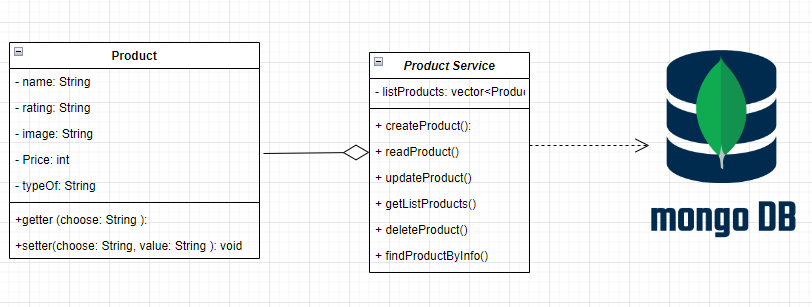
* + 1. **Admin controller**

****

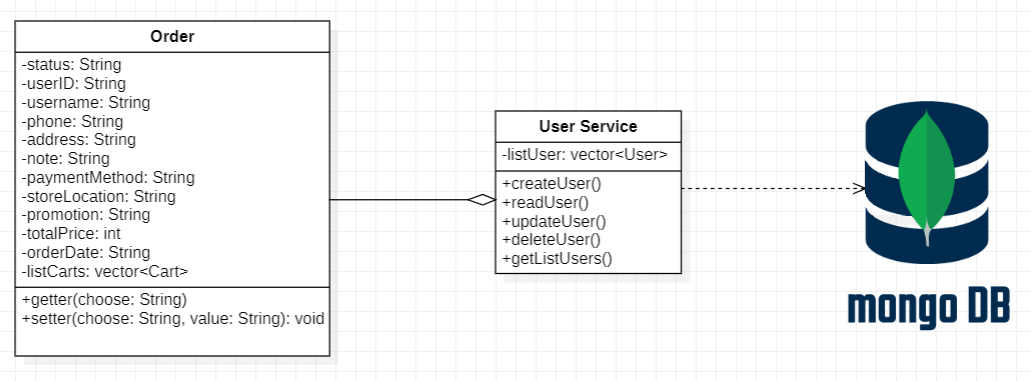
* 1. **Service**
     1. **User service**

****

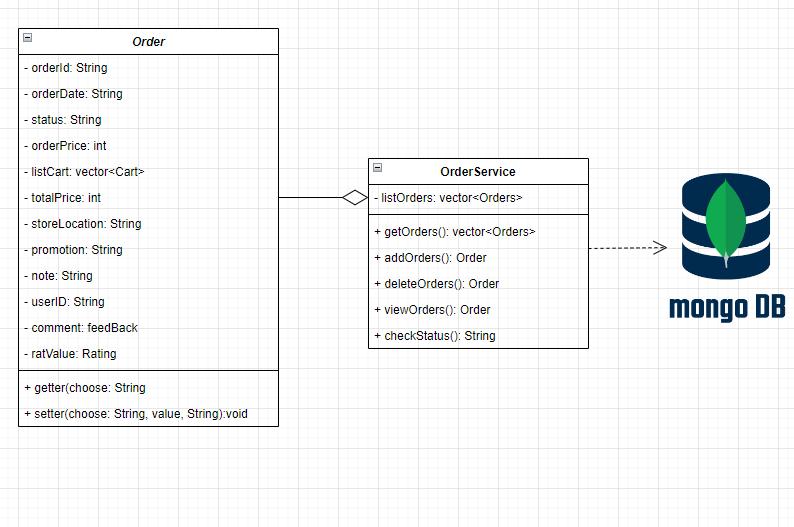
* + 1. **Product service**

****

* + 1. **Shopping cart service**

****

* + 1. **Order service:**

****

# Deployment

# Implementation View