**Лабораторна робота №4**

**Застосування патернів програмування**

1. Аналізуючи діаграми класів (Лабораторна робота №3), обґрунтувати застосування підібраних для реалізації патернів

**Архітектура**

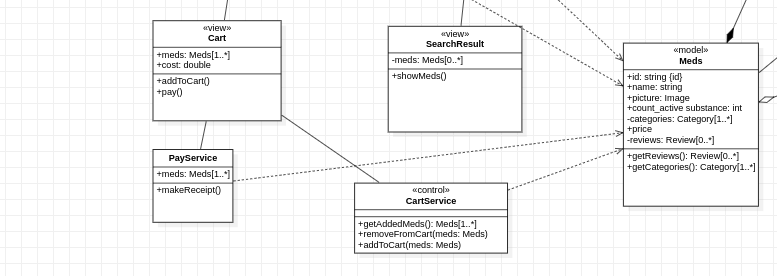
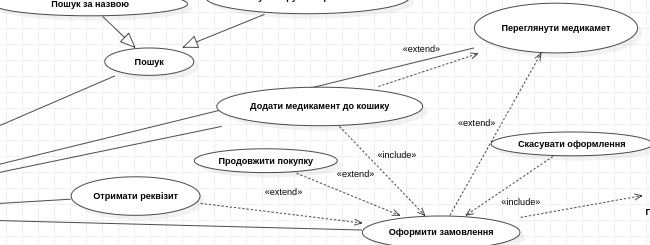
Для початку визначимо архітектуру програмного забезпечення. Потрібно створити **веб-сайт аптеки**, окрім того зазначимо, що наш замовник хоче мати систему для зберігання інформації, а саме базу даних. Враховуючи це — найкраще використати **трирівневу архітектуру “клієнт/сервер/база даних”**.

**Архітектурні патерни**

Оскільки ПЗ повино бути веб-сайтом, то доцільно буде використати **патерн MVC** (**Model-View-Controller**). Враховуючи, що нам потрібно мати доступ до бази даних доцільно використати **патерн Repository** або **DAO** (**об’єкт доступу до даних**).

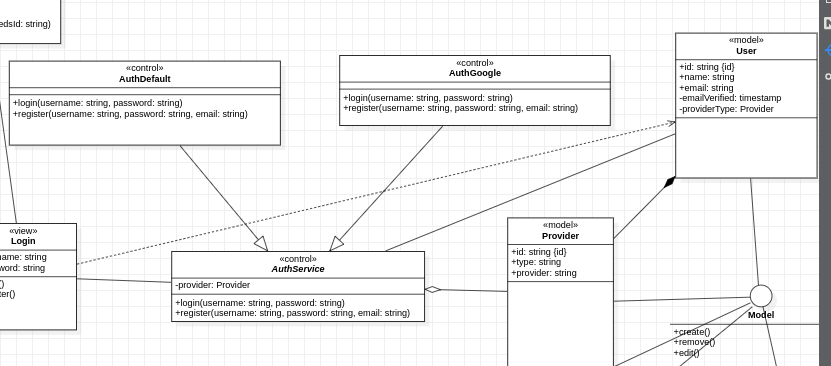
**Патерни для бекенду (Business logic)**

1. Розглянемо з Лабораторної роботи № 2 та 3, UseCase Diagram та Class Diagram, таку бізнес логіку як кошик та додавання медикаментів до нього.



Для реалізації кошика та купівлі медикаментів доцільно використати два патерни, а саме **Observer** та **State**. **Observer** — для стеження за вмістом кошику та реагування на подій, такі як додати медикамент, видалити медикамент, здійснити покупку. **State** — для зміни поведінки об’єкту залежності від стану не тільки у кошику, але й для прикладу чи є у наявності товар чи ні.

2. Авторизація різними методами (пароль+логін, через google або apple акавнт)



Тут доцільно викорситати **Decorator**, оскільки нам потрібно динамічно додавати нову функціональність для авторизації.

3. Для контролерів буде доцільно використати породжувальний патерн **Singelton** (для прикладу контролер який відповідає за медикамент повинен існувати лише один у ПЗ), а для об’єктів (моделів) **Builder**, для спрощення створення цих об’єктів та визначення динамічно їхні характеристики.

**Рекомендації до рефакторингу**

Враховуючи, що у класах **\*Service** є певна методи з однаковою логікую, які відрізняються лише об’єктом, доцільно буде створити шаблоний інтерфейс **Service**.

